Attainable Unconditional Security for Shared-Key Cryptosystems

Fabrizio Biondi, Thomas Given-Wilson, Axel Legay

Attainable Unconditional Security for Shared-Key Cryptosystems[☆]

Fabrizio Biondi[a,*], Thomas Given-Wilson[a,*], Axel Legay[a]

[a]*Inria, Campus de Beaulieu, 263 Avenue du Général Leclerc, 35042 Rennes, France*

Abstract

Preserving the privacy of private communication is a fundamental concern of computing addressed by encryption. Information-theoretic reasoning models unconditional security where the strength of the results does not depend on computational hardness or unproven results. Usually the information leaked about the message by the ciphertext is used to measure the privacy of a communication, with *perfect secrecy* when the leakage is 0. However this is hard to achieve in practice. An alternative measure is the equivocation, intuitively the average number of message/key pairs that could have produced a given ciphertext. We show a theoretical bound on equivocation called *max-equivocation* and show that this generalizes perfect secrecy when achievable, and provides an alternative measure when perfect secrecy is not achievable. We derive bounds for max-equivocation for *symmetric* encoder functions and show that max-equivocation is achievable when the entropy of the ciphertext is minimized. We show that max-equivocation easily accounts for key re-use scenarios, and that large keys relative to the message perform very poorly under equivocation. We study encoders under this new perspective, deriving results on their achievable maximal equivocation and showing that some popular approaches such as Latin squares are not optimal. We show how unicity attacks can be naturally modeled, and how relaxing encoder symmetry improves equivocation. We present some algorithms for generating encryption functions that are practical and achieve $90-95\%$ of the theoretical best, improving with larger message spaces.

Keywords: unconditional security, perfect secrecy, entropy, max-equivocation, private-key cryptography, symmetric encryption

[☆]An earlier version of this paper appeared in the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (2015)

[*]Corresponding author

Email addresses: `fabrizio.biondi@inria.fr` (Fabrizio Biondi), `thomas.given-wilson@inria.fr` (Thomas Given-Wilson), `axel.legay@inria.fr` (Axel Legay)

1. Introduction

Preserving privacy of private communication is a fundamental concern of computer science. Modern efforts can be divided into two categories: computational and unconditional security. Computational security of the privacy of a message depends on the assumed superpolynomial lower bound on complexity of some particular functions, e.g. factorization. Such lower-bound results are currently unproven, and may be weakened by technological progress in engineering, algorithmic theory, or quantum computing. Unconditional security is based on information-theoretic reasoning and proven independently of computational hardness. For this reason, unconditional security results are more general and solid than computational security results. However, the strict requirements to obtain unconditionally-secure cryptographic algorithms can make them generally impractical.

The first formal results on unconditional security were published by Shannon [10] using results from the formal theory of communication that he had just invented [9]. Shannon considers the transmission of an encrypted message on a public channel between a sender `A` and a receiver `B` that share a cryptographic key. Shannon investigates how long the key has to be to transmit the message with *perfect secrecy*, meaning that an eavesdropper `E` intercepting the encrypted message obtains no information about the original message. Shannon proves that to achieve perfect secrecy the key must be as long as the message and can be used only once, and that the one-time pad algorithm achieves this under uniform key distribution. This means that, to transmit each n-bit long message, the sender and receiver have to have previously agreed on a fresh n-bit long key, which is often impractical. Perfect secrecy is proven to be unachievable with a shared key shorter than the message to be transmitted.

In this work we consider the same scenario in which a key with a different size to the message is shared between `A` and `B`, and an encoder function using the key is used to encrypt the message into a ciphertext that is transmitted by `A` to `B` on an insecure channel. To ensure that the ciphertext is uniquely decipherable by `B` using the key, the encoder function must be injective when the key is fixed. We call this property *semi-injectivity* on the message. Such encoders are *asymmetric* in that they only require semi-injectivity on the message. However, many popular techniques and prior work consider *symmetric* encoders that also require semi-injectivity on the key.

We consider how many bits of message can be sent while respecting a suitable definition of unconditional security. However, we want to define a security condition that is also attainable when the key is smaller than the message, since this is the most common case in practice.

Since perfect secrecy cannot be achieved in this scenario, we define the maximum achievable security and how to attain it. Secrecy, as defined by Shannon, is a measure based on mutual information, and measures how much information is leaked by the communication, but not how hard it is for the attacker to decrypt the message given this leaked information. Instead of measuring secrecy, we measure the message *equivocation* of the encryption, measured as the conditional entropy of the message given the ciphertext. Equivocation, introduced by Shannon, measures how difficult it is for the attacker to decrypt the message after intercepting the ciphertext. Intuitively, *equivocation measures the average number of message/key pairs that could have produced a given ciphertext.*

We show a theoretical upper bound on equivocation, and we call this condition *max-equivocation*. We show that *max-equivocation generalizes perfect secrecy*, corresponding to perfect secrecy when the latter can be achieved and providing a characterization of maximum possible security when perfect secrecy cannot be achieved.

We derive upper and lower bounds to equivocation. For symmetric encoders we show that max-equivocation is achieved when the entropy of the ciphertext is minimized, contrarily to intuition. That is, the best theoretical encryption scheme is one that minimizes the entropy of the ciphertext.

We discuss key re-use through the perspective of max-equivocation and show that key re-use is easily accounted for. Further, that max-equivocation can be applied in key re-use scenarios to consider when a key is reaching an unacceptable level of security and so should be changed.

We show that having much more entropy in the key than the message immediately loses this extra entropy the first time the key is used. This indicates that using excessively large keys is wasteful, and instead it is better to use parts of a large key as smaller keys to achieve better overall equivocation.

We then consider encryption functions under this new perspective and show that in general the theoretical best is not achievable. We present necessary and sufficient conditions on the probability distribution over the messages for max-equivocation to be achievable. Further, we show that popular approaches to encryption functions, such as Latin squares [3, 7], are also not practically optimal.

One potential weakness of cryptosystems is the *unicity* problem [10], i.e. the fact that the eavesdropper could use the redundancy in the language of the message to gain information. We show how the unicity problem can be addressed by considering a message space divided into valid messages with positive probability and invalid messages with probability 0. Further, we show that asymmetric encoder functions can better account for invalid messages.

We revisit the prior results on symmetric encoder functions for asymmetric encoder functions, and show that they can still be derived in a meaningful manner. However, finding the theoretical max-equivocation for a given message and key distribution is a more complex optimization problem.

We present some algorithms for generating symmetric encoder functions from the message and key information that are practical and achieve reasonably good results. In general these give solutions with a quality of $90-95\%$ of the theoretical best when the message space is less than 2^9, and the quality improves as the message space increases in size.

We test different refinements to the best performing algorithm, achieving a few percentage points worse than the base algorithm for quality, in as little as approximately 55% of the runtime.

The structure of the paper is as follows. Section 2 recalls key concepts. Section 3 introduces max-equivocation and theoretic bounds. Section 4 explores key re-use and when the key has more entropy than the message. Section 5 presents results on when the theoretic bounds can and cannot be achieved. Section 6 addresses the unicity problem by considering messages with 0 probability. Section 7 considers allowing multiple keys to map the same message to the same ciphertext. Section 8 introduces and evaluates some heuristic algorithms for finding encryption functions in practice. Section 9 draws conclusions.

Related Work

This paper is an extended version of [2]. The content has been extended considering feedback received on the previous version. New material includes Sections 4, 5.4, 6, 7, 8.1, and 8.5.

Other works have considered variations or alternatives to perfect secrecy in shared-key cryptosystems. Csiszár and Körner [4] introduce *ϵ-secrecy* as a relaxing of the requirement of perfect secrecy to $I(M;C) \leq \epsilon$ for a given ϵ. Ho et al. [5] consider the bounds required for perfect secrecy in the sense of Shannon, while considering key reuse, and parameterize by the entropy of the message space. Russell and Wang [8] consider *entropic security* as an alternative, that considers min-entropy bounded statistical security measures and smaller key spaces.

The design of encryption functions, or their components such as **S**-boxes, has been widely studied [13, 7, 14], particularly when designing "optimal systems". The results here suggest that when the message space (or inputs to an **S**-box) are non-uniformly distributed then the result cannot ensure max-equivocation. Further, finding "optimal" functions should not be limited merely to Latin squares/rectangles, since higher max-equivocation can be achieved without such tight restrictions.

2. Background

This section recalls definitions and concepts from the literature pertinent to this paper. In particular we recall and define terminology in a manner useful for the results of this work.

We denote with $|\mathcal{S}|$ the size of set $\mathcal{S}$. Given sets $\mathcal{A}$ and $\mathcal{B}$ we say that a function $f : \mathcal{A} \to \mathcal{B}$ is *injective* if $\forall a_1, a_2 \in \mathcal{A}$ it holds that $f(a_1) = f(a_2) \Rightarrow a_1 = a_2$. Further, we shall denote with $\vec{a}$ the sequence $a_1, \ldots, a_i$.

2.1. Probability and Entropy

A probability distribution is used to model the (partial) knowledge of an agent about a fact, e.g. the content of a secret message being transmitted by another agent. Let X be a discrete *random variable* taking values from a *sample space* $\mathcal{X}$. A *probability distribution* on X is a function $\rho_{\mathcal{X}} : X \to [0,1]$ such that $1 = \sum_{x\in\mathcal{X}} \rho(x)$. We will just write $\rho(X)$ when the sample space is evident from the context. We write $\rho(X,Y)$ for the *joint probability distribution* on

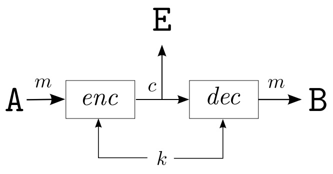

Figure 1: Shared-key cryptosystem model.

two random variables X and Y on sample spaces $\mathcal{X}$ and $\mathcal{Y}$ respectively. The *conditional probability* of X given Y is $\rho(X|Y) = \frac{\rho(X,Y)}{\rho(Y)}$. Two random variables X and Y are said to be *independent* of for each $x \in X$, $y \in Y$ it holds that $\rho(x,y) = \rho(x)\rho(y)$.

Definition 1. *Given two random variables X and Y with probability distributions $\rho(X)$ and $\rho(Y)$ and joint probability distribution $\rho(X,Y)$ we define the following non-negative real-valued functions:*

Entropy $H(X) = -\sum_{x\in X} \rho(x) \log_2 \rho(x) \quad ;$

Joint entropy $H(X,Y) = -\sum_{x\in X} \sum_{y\in Y} \rho(x,y) \log_2 \rho(x,y) \quad ;$

Conditional entropy $H(X|Y) = -\sum_{x\in X} \sum_{y\in Y} \rho(x,y) \log_2 \rho(x|y)$
$=\sum_{y\in Y} H(X|Y=y) \quad ;$

Mutual information $I(X;Y) = \sum_{x\in X} \sum_{y\in Y} \rho(x,y) \log_2 \left(\frac{rho(x,y)}{\rho(x)\rho(y)} \right)$

The generalization to more than two variables is straightforward. Note that $H(X,Y) = H(X) + H(Y)$ iff X and Y are independent.

2.2. Shared-Key Cryptosystems

A shared-key cryptosystem can be defined as follows.

Definition 2. *A* (shared-key) cryptosystem *is a 3-tuple* $(\mathcal{M}, \mathcal{K}, \mathrm{enc})$ *where:*

- *the* message space $\mathcal{M}$ *is a finite set of possible messages;*
- *the* key space $\mathcal{K}$ *is a finite set of possible keys;*
- *the* encoder enc *is a function* $\mathcal{M} \times \mathcal{K} \to \mathcal{C}$ *to some space* $\mathcal{C}$ *such that* $\forall k \in \mathcal{K}.\ \mathrm{enc}(\cdot, k)$ *is injective.*

This definition is slightly weaker than that of Biondi et al. [2] where the additional condition for the *enc* function that "$\forall m \in \mathcal{M}.\ enc(m, \cdot)$ is injective" was also required. The additional condition can simplify some results, although we show that removing the additional condition can strengthen others. We highlight these differences when the results are presented later in the paper.

A shared-key cryptosystem induces the set of its possible ciphertexts and a decoder function.

Definition 3. *Let $\mathcal{C} = \{c \mid \exists m \in \mathcal{M}.\ \exists k \in \mathcal{K}.\ c = enc(m, k)\}$ be the* ciphertext space*, i.e. the set of possible ciphertexts c given $\mathcal{M}$, $\mathcal{K}$ and enc.*

Definition 4. *Let the* decoder dec *be the function $\mathcal{C} \times \mathcal{K} \to \mathcal{M}$ such that $dec(enc(m, k), k) = m$.*

The existence and uniqueness of such a decoder function is ensured by the requirement that $enc(\cdot, k)$ is injective.

The channel model of a cryptosystem was introduced by Shannon [10] as depicted in Figure 1. In this model, agent `A` wants to send a message $m \in \mathcal{M}$ to agent `B` on a public channel that is eavesdropped by agent `E`. Initially, `A` and `B` share a secret *key* $k \in \mathcal{K}$.

`A` encodes the message m with key k using an appropriate encoder function *enc*, obtaining the *ciphertext* c. `A` sends c to `B` via a public channel, where c is also read by the eavesdropper `E`. Knowing the key k, `B` decodes the message m using the inverse of the encoding function, *dec*. Not knowing the key k, `E` tries to infer m from their knowledge of *enc* and c and using unlimited computational power.

Consider the information available to agent `E`. Agent `E` knows the encoder function *enc*, and ciphertext c, but not the message m or key k. We will use random variables to model `E`'s knowledge about the communication before and after `E` intercepts the ciphertext c. Let M (resp. K, C) be a random variable on the support set $\mathcal{M}$ (resp. $\mathcal{K}$, $\mathcal{C}$) representing the *a priori* value of the message m (resp. key k, ciphertext c) according to `E`, i.e. the value before the ciphertext is intercepted.

Random variables M and K are assumed to be independent, so

$$H(M, K) = H(M) + H(K)\ .$$

We call $H(M)$ the *prior entropy* of M, modeling the amount of uncertainty that `E` has on the message before observing the ciphertext. Similarly, we call $H(M|C)$ the *posterior entropy* of M given C, modeling the uncertainty left on the message after observing the ciphertext. The prior entropy gives a measure of how hard it is for `E` to guess the message before observing the ciphertext, while the posterior entropy measures the same after observing the ciphertext.

Shannon defined *perfect secrecy* as the highest possible security condition attainable on a cryptosystem [10]. Perfect secrecy is attained when the mutual information between the message and the ciphertext is zero; when knowledge of the ciphertext gives no information about the message. That is, the prior and posterior entropies coincide, meaning that observing the ciphertext did not reduce the uncertainty of `E` on the message in any way.

Definition 5 (Perfect Secrecy)**.** *A cryptosystem attains* perfect secrecy *iff*

$$H(M) = H(M|C)$$

It can be shown that

$$H(M) = H(M|C) \Leftrightarrow I(M; C) = 0$$

thus perfect secrecy is attained when the mutual information between the message and the ciphertext is 0.

Shannon also proved that for a cryptosystem to be perfectly secure it is necessary that the key space is at least as large as the message space, i.e.

$$|\mathcal{K}| \geq |\mathcal{M}|$$

making perfect secrecy hard to achieve in practice. Indeed, Shannon proved that perfect secrecy can be achieved when $|K| = |M|$ as long as the keys are uniformly distributed [10]. Observe that such requirements induce the relation that

$$H(K) \geq H(M) \ .$$

However, this requires `A` and `B` to have exchanged such a key k before sending the message m, and k can be used only once.

2.3. Latin Squares and Rectangles

A *Latin rectangle* is a $a \times b$ matrix with elements in $\{1, \ldots, b\}$ such that all the elements in each row and each column are distinct. It is a *Latin square* iff $a = b$.

An encoder function *enc* can be represented as a Latin rectangle if $|C| = \max(|M|, |K|)$ and $\forall m \in \mathcal{M}.\ enc(m, \cdot)$ is injective. This takes the form of a $|K| \times |M|$ matrix with elements in $\{c_1, \ldots, c_{|C|}\}$ [10, 3]. However, we will show in Section 5 that an optimal encoder function is in general not a Latin rectangle.

2.4. Different Entropy Measures

Recently it has been shown that Shannon entropy measures the effort for decrypting the ciphertext by asking arbitrary binary questions, which is not a general security scenario [6]. Other measures based on different entropies have been introduced. Smith's *min-entropy* [11] has been shown to measure the resistance of the cryptosystem against one-time attacks in which the attacker has a single chance to guess the message correctly. Other entropy measures include g-leakage, that parametrizes min-entropy with a given gain function [1], and guessing entropy, that quantifies the effort required to decrypt the ciphertext with equality tests [6]. In particular, the gain functions used in g-leakage computation can be used to model complex information about the message spaces, such as some messages being more valuable than others.

The results presented in this paper can be re-derived using any other entropy measure, generating alternate definitions of max-equivocation that consider different types of attackers. Modification of the heuristics to use a different type of entropy is straightforward. In general, different encoder functions can be optimal for different entropy measures.

Further discussion about the effect of choosing different measures of entropy is not in the scope of this paper.

3. Max-Equivocation

We want to provide a measure of the security of a shared-key cryptosystem that is meaningful whether or not the the key used is smaller than the message to be sent.

Consider the entropy $H(M)$ of the message m. It measures the lack of information that `E` has about the message, so a higher $H(M)$ corresponds to a greater difficulty for `E` to guess the message. In our scenario, `A` and `B` share a secret key k, and the lack of information that `E` has about k is measured by $H(K)$. The amount of entropy left in the message after the attacker has intercepted the ciphertext is measured by the message equivocation $H(M|C)$, so we focus upon this.

Shannon proved that $H(K) \geq H(M)$ is a necessary but not sufficient condition to find an encoder function *enc* such that $I(M;C) = 0$. Since $I(M;C) = H(M) - H(M|C)$, this corresponds to $H(M|C) = H(M)$, thus the message equivocation is equivalent to the entropy of the message. Since $H(K) \geq H(M)$, it is also true that $H(M|C) \leq \min(H(M), H(K))$.

Consider the complementary case in which $H(K) < H(M)$. Since `E` knows every other detail of the communication except for k, we will show that it is not possible to encrypt m as a ciphertext $c \in C$ in a way such that the message equivocation $H(M|C)$ is greater than the entropy $H(K)$, i.e. $H(M|C) \leq H(K)$. Since $H(K) < H(M)$, this again corresponds to $H(M|C) \leq \min(H(M), H(K))$.

We conclude that irrespective of the relative sizes of the entropies of the message space and key space, the theoretical maximum equivocation is always achieved when it corresponds to their minimum. We say that a cryptosystem respects *max-equivocation* when it achieves this upper bound.

Definition 6. *A cryptosystem achieves* max-equivocation *iff*

$$H(M|C) = \min(H(M), H(K)) \ .$$

Key equivocation can similarly be defined as $H(K|C)$, measuring the average number of keys that could have produced a given ciphertext.

We will now introduce semi-injectivity, the core property of an encoder function. We will show that as a consequence of the encoder function being semi-injective on both message and key, message equivocation and key equivocation actually coincide, so we can just call them equivocation.

3.1. Semi-Injectivity

We introduce the concept of *semi-injectivity* of a function, that will be used throughout the paper.

Definition 7. *A function $f : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}$ is* semi-injective *on $\mathcal{B}$ iff it holds that*

$$\forall a \in \mathcal{A},\ \forall b_i, b_j \in \mathcal{B}.\ f(a, b_i) = f(a, b_j) \Leftrightarrow b_i = b_j \ .$$

Observe that a function on a tuple can be semi-injective on any element of the tuple, and that a semi-injective function on a single argument is injective. Semi-injectivity is equivalent to the property that the function can be Curried and after being applied to one argument the remaining function is injective, i.e. $f : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}$ is semi-injective on $\mathcal{B}$ when $f' : \mathcal{A} \rightarrow \mathcal{B} \rightarrow \mathcal{C}$ and $f'(a) : \mathcal{B} \rightarrow \mathcal{C}$ is injective.

Lemma 1. *Let a function $f : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}$ be semi-injective on $\mathcal{A}$. The image $f[\mathcal{A}, \mathcal{B}]$ of f in $\mathcal{C}$ has at least the size of $\mathcal{A}$:*

$$|f[\mathcal{A}, \mathcal{B}]| \geq |\mathcal{A}| \ .$$

Consequently, if f is semi-injective in both $\mathcal{A}$ and $\mathcal{B}$ then

$$|f[\mathcal{A}, \mathcal{B}]| \geq \max(|\mathcal{A}|, |\mathcal{B}|) \ .$$

Lemma 2. *Let $f : \mathcal{A} \times \mathcal{B} \rightarrow \mathcal{C}$ be semi-injective on $\mathcal{A}$, A be a random variable on $\mathcal{A}$, $\rho_{\mathcal{A}}(A)$ a probability distribution on A, and $H(A)$ the entropy of $\rho_{\mathcal{A}}(A)$. Similarly, let C be a random variable on $f[\mathcal{A}, \mathcal{B}]$, $\rho_{\mathcal{C}}(C)$ a probability distribution on C, and $H(C)$ the entropy of $\rho_{\mathcal{C}}(C)$. Then*

$$H(A) \leq H(C) \ .$$

Proof. Recall that $|f[\mathcal{A}, \mathcal{B}]| \geq |\mathcal{A}|$ by Lemma 1. If $|f[\mathcal{A}, \mathcal{B}]| = |\mathcal{A}|$ then there must be one $c \in f[\mathcal{A}, \mathcal{B}]$ that corresponds to each $a \in \mathcal{A}$ and thus $H(A) \leq H(C)$. If $|f[\mathcal{A}, \mathcal{B}]| > |\mathcal{A}|$ then there must be at least two c_i and c_j such that $f(a, b_x) = c_i$ and $f(a, b_y) = c_j$. This increases the entropy, due to either $H(B)$ (when $b_x \neq b_y$), or via f itself otherwise. In either case, $H(A) \leq H(C)$. □

Note that Lemma 2 can also be proved by concavity of H.

Definition 8. *An encoder function $enc : \mathcal{M} \times \mathcal{K} \rightarrow \mathbb{N}$ is a* symmetric encoder function *iff it is semi-injective on $\mathcal{M}$ and $\mathcal{K}$.*

Definition 8 of a symmetric encoder function is the same as an "encoder function" of [2]. The rest of this section shall consider only symmetric encoder functions (Definition 8) since these simplify the results.

Definition 9. *A function $enc : \mathcal{M} \times \mathcal{K} \rightarrow \mathbb{N}$ is an* (asymmetric) encoder function *iff it is semi-injective on $\mathcal{M}$.*

Definition 9 here is to generalize results and will be highlighted later in the paper. Later in Section 7 this is relaxed to consider encoder functions in general (Definition 9).

Since the encoder function is semi-injective on $\mathcal{M}$, then the message is uniquely determined by a given ciphertext c and key k, thus $H(M|K, C) = 0$. Analogous reasoning on a symmetric encoder being semi-injective on $\mathcal{K}$ shows that $H(K|M, C) = 0$. We use these results to show that message equivocation and key equivocation coincide for symmetric encoder functions.

Theorem 1. $H(M|C) = H(K|C)$.

Proof. The proof adapts the similar proof of Theorem 1 of [12, sec. 7.3]:

$$\begin{aligned}
H(M|C) &= H(M,C) - H(C) \\
&= H(M,K,C) - H(K|M,C) - H(C) \\
&= H(M,K,C) - H(M|K,C) - H(C) \\
&\qquad\qquad (\text{since } H(M|K,C) = H(K|M,C) = 0) \\
&= H(K,C) - H(C) = H(K|C) \ .
\end{aligned}$$

□

Since message equivocation and key equivocation coincide for symmetric encoders by Theorem 1, we will just refer to them as equivocation, denoted by $H(M|C)$.

3.2. Bounds on Equivocation

This section is dedicated to studying upper and lower bounds for equivocation, and to find necessary and sufficient conditions for a cryptosystem to achieve max-equivocation.

Consider after agent `E` intercepts the ciphertext c, their *a posteriori* information about the message is measured by the equivocation $H(M|C)$.

Lemma 3. *The entropy of the ciphertext has: lower bound of the greater of the entropy of the message or the key; and upper bound of the entropy of the message plus the entropy of the key. That is,*

$$\max(H(M), H(K)) \leq H(C) \leq H(M) + H(K) \ .$$

Proof. By Definition 8 and Lemma 1. □

Lemma 4. *The entropy of the ciphertext given the message is that of the key. That is, $H(C|M) = H(K)$.*

Proof. By Definition 8 of *enc* being symmetric and thus semi-injective on K and Lemma 2. □

Theorem 2. *Equivocation ranges from 0 to the minimum of the entropy of the message and the entropy of the key, i.e.*

$$0 \leq H(M|C) \leq \min(H(M), H(K)) \ .$$

Proof.

$$\begin{aligned}
H(M|C) &= H(C|M) + H(M) - H(C) \\
&= H(K) + H(M) - H(C) \qquad\qquad (\text{by Lemma 4})
\end{aligned}$$

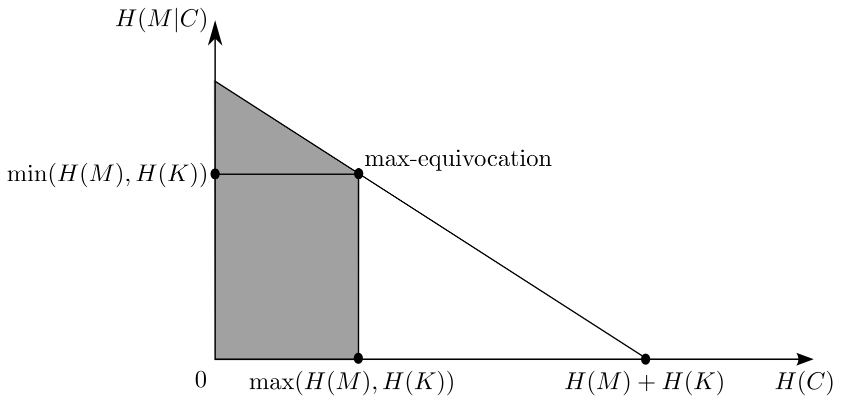

Figure 2: Graph showing equivocation $H(M|C)$ as a function of the entropy of the ciphertext $H(C)$. The gray area is unachievable, since $H(C) \geq \max(H(M), H(K))$ by Lemma 3, thus the maximum possible equivocation is $H(M|C) = \min(H(M), H(K))$.

with the lower bound being

$$\begin{aligned} H(M|C) &\geq H(K) + H(M) - (H(M) + H(K)) && \text{(by Lemma 3)} \\ &= 0 \end{aligned}$$

and upper bound

$$\begin{aligned} H(M|C) &\leq H(K) + H(M) - (\max(H(M), H(K))) && \text{(by Lemma 3)} \\ &= \min(H(K) + H(M) - H(M), \\ &\qquad H(K) + H(M) - H(K))) \\ &= \min(H(K), H(M)) \ . \end{aligned}$$

□

Corollary 1. *A necessary and sufficient condition for max-equivocation is that the entropy of the ciphertext is equal to the maximum between the entropy of the key and the entropy of the message:*

$$\begin{gathered} H(C) = \max(H(K), H(M)) \\ \Updownarrow \\ H(M|C) = \min(H(K), H(M)) \ . \end{gathered}$$

Proof. Recall that

$$\begin{aligned} H(M|C) &= H(C|M) + H(M) - H(C) \\ &= H(K) + H(M) - H(C) \ \ . && \text{(by Lemma 4)} \end{aligned}$$

Since $H(M)$ and $H(K)$ are constant and $H(C)$ is negative in the equation, then $H(M|C)$ is maximal iff $H(C)$ is minimal. □

Since by Lemma 3 we know that $H(C) \geq \max(H(K), H(M))$, this means that *max-equivocation is achieved when the entropy of the ciphertext is minimal.* The relation between equivocation and entropy of the ciphertext is depicted in Figure 2.

Note that the idea that the entropy of the ciphertext has to be *minimized* is not intuitive, and claims of the opposite can be found e.g. in [12, p. 117]. In the next Sections we will study how to construct a symmetric encoder function minimizing the entropy of the ciphertext, and we will show that the theoretical minimum of $H(C) = \max(H(K), H(M))$ cannot always be achieved.

4. Key Re-Use and Key Equivocation

This section considers key re-use, and how the results on equivocation can account for such scenarios. This also leads into considering the impact on key equivocation, and scenarios where the key has more entropy than the message.

4.1. Key Re-Use

The results so far have focused on when a single message is sent with a single key, however, the results can be generalized to account for scenarios where multiple messages are sent with the same key. This section shows how this can be easily accounted for.

Consider the sequential transmission of a vector of n independent messages $\vec{m} \in \mathcal{M}^n$ chosen according to the same prior distribution ρ such that $\rho(M)$ has entropy $H(M)$. Also consider a vector of keys $\vec{k} \in \mathcal{K}^n$, not necessarily independent. Each transmission uses the same encoder *enc*. Transmission i encodes message m_i with key k_i producing ciphertext c_i. The prior entropy of the messages is $H(\vec{M}) = nH(M)$ since the elements of $\vec{m}$ are independent. The equivocation of the repeated transmission is

$$\begin{aligned} H(\vec{M}|\vec{C}) &= H(\vec{C}|\vec{M}) + H(\vec{M}) - H(\vec{C}) \\ &= H(\vec{C}|\vec{M}) + nH(M) - H(\vec{C}) \\ &= H(\vec{K}) + nH(M) - H(\vec{C}) \ . && \text{(by Lemma 4)} \end{aligned}$$

Observe that the entropy of $H(\vec{K})$ is maximized to $nH(K)$ when the keys are independent, and is minimized at $H(K)$ when the same key is used to encode all messages. Similarly, the entropy of $H(\vec{C})$ is between $\max(H(\vec{K}), H(\vec{M}))$ and $H(\vec{K}) + H(\vec{M})$ by Lemma 3, so it still holds that $H(\vec{M}|\vec{C}) \leq H(\vec{K})$. Note that the result holds whether: we use n different independent keys, in which case $H(\vec{K}) = nH(K)$; we re-use the same key n times, in which case $H(\vec{K}) = H(K)$; or we use keys with any other dependencies between them.

This shows that given a fixed amount of entropy to be used as the key in a shared-key cryptosystem, there is no difference in the equivocation when transmitting independent messages sequentially using parts of the total shared key, or repeatedly using the whole shared key for each message. In the latter case, it can be considered that the first transmission encodes a message m_1 with the key k producing ciphertext c_1 and has a maximum equivocation of $H(M_1|C_1) = H(K|C_1) \leq H(K)$ by Theorem 1. Then the second transmission encodes m_2 with k producing c_2 and the equivocation of both transmission becomes $H(M_1, M_2|C_1, C_2) = H(K|C_1, C_2) \leq H(K|C_1) \leq H(K)$, and so on.

Indeed, one could simply combine all the messages to be transmitted $\vec{m}$ into a single message as above and send this using all the shared key. Of course in practice it is far easier to do the reverse, and send parts of the message as blocks in some shared-key cipher such as AES, in which case the results here show the bounds on equivocation.

Since the entropy of the key decreases each time it is used, it is reasonable to continue using the same key until its entropy falls below some predetermined lower bound that is considered sufficient for the security of the transmission. When the entropy of the key becomes too low for the key to guarantee an acceptable equivocation, a fresh key is generated by the key exchange protocol used.

It could be thought that using a very large key could be beneficial for the key re-use scenario, since the key could be used many times before its entropy becomes too small. However, this is not necessarily the case. In fact, in the next section we will show that if a key with more entropy than the message is used, the excess entropy of the key is immediately lost. Therefore, there is no reason to use a key with more entropy than the message, specially if the key is to be used multiple times.

4.2. Bounds on Key Equivocation

Consider the total information about the message and the key. We show that the amount of information leaked about the message and the key corresponds at least to the absolute difference of their entropies for symmetric encoder functions.

We start by showing that the key equivocation $H(K|C)$ varies between 0 and the minimum of $H(K)$ and $H(M)$, just like the message equivocation does:

Theorem 3. $0 \leq H(K|C) \leq \min(H(M), H(K))$.

Proof. Since K and M are independent and *enc* is semi-injective in both, we can just rewrite the proof of Theorem 2 by substituting M with K and vice versa. □

We can now prove the main theorem of this subsection.

Theorem 4. $I(M, K; C) \geq |H(K) - H(M)|$.

Proof.

$$\begin{aligned}
I(M, K; C) &= H(M, K) - H(M, K|C) \\
&= H(M) + H(K) - H(M|C) - H(K|C) \\
&\qquad\qquad \text{(by independence of } M \text{ and } K\text{)} \\
&\geq H(M) + H(K) - 2\min(H(M), H(K)) \quad \text{(by Theorems 2 and 3)} \\
&= |H(K) - H(M)|
\end{aligned}$$

□

Theorem 4 proves that the information leakage about the message and key is minimal when they have the same size. Furthermore, this proves that having a key with more entropy than the message is unnecessary. In particular, we show that if $H(K) > H(M)$ and perfect secrecy holds, then $H(M)$ bits of the key are used to protect the message with perfect secrecy, and at least the remaining $H(K) - H(M)$ bits of the key are leaked to the attacker.

Lemma 5. *Assume $H(K) > H(M)$ and $I(M; C) = 0$. Then*

$$I(K; C) \geq H(K) - H(M) .$$

Proof.

$$\begin{aligned}
I(K; C) &= H(K) - H(K|C) \\
&\geq H(K) - \min(H(M), H(K)) && \text{(by Theorem 3)} \\
&= H(K) - H(M) && \text{(since } H(K) > H(M)\text{)}
\end{aligned}$$

□

This proves that using a key with more entropy than the message is unnecessary, since for perfect secrecy it is sufficient that the entropies of the key and the message coincide, and any additional entropy of the key is leaked.

5. Uniform Key, Non-Uniform Message

This section considers some conditions for when max-equivocation can be achieved, and examples of when max-equivocation is not achievable. In general max-equivocation is not achievable and Latin squares prove to not even be optimal with respect to max-equivocation.

5.1. Achieving Max-Equivocation

From Corollary 1 and Lemma 3 we have that the max-equivocation is ensured when the entropy of the ciphertext is minimized. Unfortunately max-equivocation (and thus perfect secrecy) cannot be achieved in all scenarios. Max-equivocation can be achieved for any distribution over the message space when the key space is of equal size and the key distribution is uniform. The solution for max-equivocation in this case is any encryption function that can be represented as a Latin square.

Observe from Corollary 1 that $H(C) = \max(H(K), H(M))$ and that since $|\mathcal{K}| = |\mathcal{M}|$ and $\mathcal{K}$ has uniform distribution that $H(K) \geq H(M)$. From this we require that $H(C) = H(K)$ for max-equivocation.

The easiest way to achieve $H(C) = H(K)$ is for $|\mathcal{C}| = |\mathcal{K}|$ and for the distribution of $\mathcal{C}$ to be uniform. This is achieved when the encryption function corresponds to a Latin square.

Theorem 5. *Given $|\mathcal{K}| = |\mathcal{M}|$ and uniform distribution over $\mathcal{K}$, any encryption function enc that can be represented as a Latin square achieves max-equivocation.*

Proof. As *enc* can be represented as a Latin square it follows that every $c \in \mathcal{C}$ must appear in each row and each column exactly once. Since the key distribution is uniform, the probability for each c is $\sum_{m \in \mathcal{M}} \frac{\rho(m)}{|\mathcal{K}|}$ and since $\sum_{m \in \mathcal{M}} \rho(m) = 1$ it follows that $\rho(c) = \frac{1}{|\mathcal{K}|}$. Since in a Latin square $|\mathcal{M}| = |\mathcal{C}| = |\mathcal{K}|$ conclude via the uniform distribution over $\mathcal{C}$ that $H(C) = H(K)$ and Corollary 1. □

5.2. Unachievable Max-Equivocation

In general max-equivocation cannot be achieved. Indeed, there are small examples where $H(C)$ cannot be made to match $\max(H(K), H(M))$. For example, consider when there is a message space $\mathcal{M} = \{m_1, m_2, m_3\}$ with distribution $\rho(M) = \{m_1 \mapsto 0.2, m_2 \mapsto 0.3, m_3 \mapsto 0.5\}$ and the key space is of size 2 with uniform distribution. It turns out that there is no encoder function such that $H(C) = H(M) \approxeq 1.4854$. The best possible is a Latin rectangle where the symbols must have the probabilities $\rho(c_1) = 0.25$ and $\rho(c_2) = 0.35$ and $\rho(c_3) = 0.4$ which yields $H(C) \approxeq 1.5588$ (checked via brute forcing all possibilities).

5.3. Non-Optimality of Latin Rectangles

When $|\mathcal{M}| \neq |\mathcal{K}|$, each Latin rectangle of size $|\mathcal{K}| \times |\mathcal{M}|$ represents an encoder function with $|\mathcal{C}| = \max(|\mathcal{M}|, |\mathcal{K}|)$. We can compute that the minimum and maximum entropy of the ciphertext space generated by such an encoder function are bounded by:

$$\max(H(M), H(K)) \leq H(C) \leq \log(\max(|\mathcal{M}|, |\mathcal{K}|)) \ .$$

When the distribution on the larger side of the rectangle is uniform then any Latin rectangle achieves max-equivocation, as shown in Section 5.1.

When the distribution on the larger side of the rectangle is not uniform, it is possible that the optimal encoder function is not a Latin rectangle. For instance, let $\mathcal{M} = \{m_1, m_2, m_3\}$ with distribution $\rho(M) = \{m_1 \mapsto 0.02, m_2 \mapsto 0.49, m_3 \mapsto 0.49\}$ and let the key be uniform on $\mathcal{K} = \{k_1, k_2\}$. The best Latin rectangle encoding shown in Table 1a has $\mathcal{C} = \{c_1, c_2, c_3\}$ and $H(C) \approxeq 1.5097$, while the optimal encoding shown in Table 1b has $\mathcal{C} = \{c_1, c_2, c_3, c_4\}$ and $H(C) \approxeq 1.1414$.

		Message		
		m_1	m_2	m_3
Key	k_1	c_1	c_2	c_3
	k_2	c_2	c_3	c_1

(a) Best Latin rectangle

		Message		
		m_1	m_2	m_3
Key	k_1	c_1	c_2	c_3
	k_2	c_4	c_3	c_2

(b) Optimal encoder

Table 1: Encoder functions for message distribution $\rho(M) = \{m_1 \mapsto 0.02, m_2 \mapsto 0.49, m_3 \mapsto 0.49\}$ and uniform key distribution.

Note that neither reaches the theoretical lower bound of $H(C) = H(M) \approxeq 1.1214$.

Due to the inherent complexity of finding an encoder function that minimizes the entropy of the ciphertext, computing an optimal encoder function for a given key and message distribution is not trivial. We introduce different heuristics and compare them experimentally in Section 8.

5.4. Conditions for Achievability

Assume $\log(|\mathcal{K}|) = H(K) < H(M)$. Then the maximum achievable message equivocation depends on the probability distribution $\rho(M)$ on the message.

We introduce the set $\mathcal{E}_c \subseteq \mathcal{M}$ of the messages that can be decoded from a given ciphertext c using any key $k \in \mathcal{K}$.

Definition 10. *Let enc be an encoder function and $c \in \mathcal{C}$ one of its ciphertextsymbols. Then the set $\mathcal{E}_c$ is*

$$\mathcal{E}_c = \{m \in \mathcal{M} \mid \exists k \in \mathcal{K}.\ enc(m,k) = c\}\ .$$

Note that $|\mathcal{E}_c| \leq |\mathcal{K}|$, with equality when the encoder is symmetric. Let E_c be a random variable over $\mathcal{E}_c$ with probability distribution

$$\forall m \in \mathcal{E}_c.\ \rho_{\mathcal{E}_c}(m) = \frac{\rho_{\mathcal{M}}(m)}{\sum_{m \in \mathcal{E}_c} \rho_{\mathcal{M}}(m)}\ .$$

The probability distribution $\rho_{\mathcal{E}_c}$ represents the distribution over the messages according to the attacker after they have intercepted the ciphertext c, and its entropy is $H(E_c) = H(\rho_{\mathcal{E}_c}(E))$. Then we can compute the message equivocation of the encoder function *enc* as the expectation of the entropy of $H(E_c)$ over all possible ciphertexts.

Lemma 6.

$$H(M|C) = \sum_{c \in \mathcal{C}} \rho_{\mathcal{C}}(c) H(E_c)\ .$$

Proof. It is sufficient to prove that $H(E_c) = H(M|C = c)$ for all $c \in \mathcal{C}$. Note that for each $c \in \mathcal{C}$ it holds that $\rho_{\mathcal{M}}(m \notin \mathcal{E}_c | C = c) = 0$. Also, for each $m \in \mathcal{M}$

it holds that

$$\rho_{\mathcal{M}}(m|C=c) = \frac{\rho_{\mathcal{M}}(m)}{\sum_{m\in\mathcal{E}_c}\rho_{\mathcal{M}}(m)} = \rho_{\mathcal{E}_c}(m)$$

and the result follows. □

This shows that maximizing message equivocation means maximizing the entropy E_c of each set $\mathcal{E}_c$. Since $|\mathcal{E}_c| \leq |\mathcal{K}|$ and $H(K) = \log(\mathcal{K})$ then $H(E_c) \leq H(K)$, thus to achieve $H(M|C) = H(K)$ we need every $\mathcal{E}_c$ to have size equal to $|\mathcal{K}|$ and the distribution $\rho_{\mathcal{E}_c}$ over it to be uniform. This justifies the following theorem.

Theorem 6. *An encoder function achieving $H(M|C) = H(K)$ exists iff the message space $\mathcal{M}$ can be partitioned into subsets $\mathcal{M}_1, \ldots \mathcal{M}_n$ such that for each $1 \leq i \leq n$ it holds that:*

1. $|\mathcal{M}_i| \geq |\mathcal{K}|$*; and*
2. *messages in $\mathcal{M}_i$ are equiprobable.*

Proof. Forward direction

Assume by contradiction that an encoder function achieving $H(M|C) = H(K)$ exists for a message space that cannot be partitioned into subsets respecting conditions 1) and 2).

Consider when there exists some partition $\mathcal{M}_i$ that has size smaller than $|\mathcal{K}|$ and all partitions $\mathcal{M}_j$ are such that $m \in \mathcal{M}_j$ are equiprobable. Now consider one ciphertext symbol c such that $\exists m \in \mathcal{M}_i$ where $enc(m,k) = c$. Now if $\mathcal{E}_c$ has size smaller than $|\mathcal{K}|$ then it follows that $H(M|C) < H(K)$. Therefore there must exist some $m' = dec(c,k)$ such that $m' \notin \mathcal{M}_i$. If $\rho(m) \neq \rho(m')$ then again it follows that $H(M|C) < H(K)$. Lastly if $\rho(m) = \rho(m')$ then it follows that m' can instead be partitioned into $\mathcal{M}_i$. To conclude; if $|\mathcal{M}_i \cup \{m'\}| = |\mathcal{K}|$ this violates the premise, otherwise there must be some other m's that eventually leads to $|\mathcal{M}_i \cup \{m'\} \cup \{m''\} \cup \ldots| \geq |\mathcal{K}|$ and thus contradiction.

Consider when there exists some partition $\mathcal{M}_i$ that contains m_1 and m_2 such that $\rho(m_1) \neq \rho(m_2)$. Now if there exists some c and k_1 and k_2 such that $enc(m_1,k_1) = c = enc(m_2,k_2)$ then it follows that $H(M|C) < H(K)$. Therefore for $H(M|C) = H(K)$ to hold there must be $|\mathcal{K}| - 1$ other m_i's with $\rho(m_i) = \rho(m_1)$ and also m_j's with $\rho(m_j) = \rho(m_2)$. It follows that $\mathcal{M}$ can be partitioned to have these sets of m's in different partitions achieving the conditions.

Reverse direction

We produce an encoder function achieving $H(M|C) = H(K)$. Consider a single partition $\mathcal{M}_i$. Since the distribution over the messages in $\mathcal{M}_i$ is uniform then any Latin rectangle encoder for $\mathcal{M}_i$ achieves $H(M_i|C) = H(K)$ by Theorem 5. By producing a Latin encoder for each partition and concatenating them we obtain an encoder achieving $H(M|C) = H(K)$. □

6. Unicity

This section addresses the unicity problem for max-equivocation by considering the message space to contain both valid and invalid messages.

6.1. Motivation

In this section we will consider the messages as represented by sequences of symbols in some language, e.g. letters in the English language. In general not all sequences of symbols in the language represent a (valid) message that the sender may wish to transmit. For instance, the letter sequence "snfeigkpangrj" has no meaning in English, i.e. is not "valid" English.

Recall that having access to unlimited computational power, the eavesdropper `E` can list all possible messages that could have been encoded as a given ciphertext. However, many of these possible messages may be sequences of letters that have no meaning in English, and thus are not valid messages. Thus, a viable form of attack for `E` is to decrypt a ciphertext with all possible keys and keep the only possible valid messages. This can vastly reduce the equivocation of the message. The longer a sequence of letters is, the less likely it is to be a valid message in the given language. In particular, according to the language, there is a certain message length such that we expect only one of the messages decodable from the ciphertext to be valid. This is a common problem in languages that are highly structured where many sequences of symbols do not represent valid messages, e.g. English; this is referred to as *redundancy* of the language.

This is the core intuition behind the unicity problem and unicity based attackers, for a thorough and formal treatment of this we refer to [10, 12].

In this section we consider what happens of the message space, e.g. sequences of letters, is divided into valid and invalid messages, e.g. sequences that do and do not correspond to meaningful messages in the English language. We show that the results of this paper still apply by considering invalid messages as having probability 0 in the message distribution.

6.2. Valid and Invalid Messages

Consider a set $\mathcal{M}_v$ of messages among which the sender chooses the one they want to transmit to the receiver: we will call these the *valid messages.*

Fix an alphabet Σ that will be used for the representation of the messages, also fix a length l such that $|\Sigma^l| \geq |\mathcal{M}_v|$. Now pick an injection from $\mathcal{M}_v$ into Σ^l and thus yield the sequences of Σ^l that represent the valid messages. It follows that there are $|\Sigma^l| - |\mathcal{M}_v|$ sequences in Σ^l that do not correspond to any valid message in $\mathcal{M}_v$; we will call the set of these sequences the *invalid messages* $\mathcal{M}_i$. Since the message encoding can exploit compression to a different language and alphabet (e.g. binary) or other techniques, we can assume that $0 \leq |\mathcal{M}_i| < |\mathcal{M}_v|$ without loss of generality.

In Section 5 no assumption was made upon the alphabet of the messages, thus allowing $\log_{|\Sigma|} |\mathcal{M}_v| \in \mathbb{N}$ and consequently $\mathcal{M}_i = \emptyset$. In this section we consider vulnerabilities that can arise when the set of invalid messages is not empty.

Let $\mathcal{M} = \mathcal{M}_v \cup \mathcal{M}_i$ be the set of all messages. The probability distribution $\rho_{\mathcal{M}_v}(M_v)$ on random variable M_v on set $\mathcal{M}_v$ can be extended to a probability distribution $\rho_{\mathcal{M}}(M)$ on random variable M on set $\mathcal{M}$ as

$$\rho_{\mathcal{M}}(m) = \begin{cases} \rho_{\mathcal{M}_v}(m) & \text{if } m \in \mathcal{M}_v \\ 0 & \text{otherwise} \end{cases}$$

meaning that invalid messages are part of the message space but they have probability 0 of being chosen by the sender as the message to be sent. Note that $H(M) = H(M_v)$.

Here we assume a symmetric encoder function $enc : \mathcal{M} \times \mathcal{K} \to \mathbb{N}$ with image $\mathcal{C}$. By its semi-injectivity on $\mathcal{M}$ it holds that each ciphertext $c \in \mathcal{C}$ is the image of at most $|\mathcal{K}|$ messages in $\mathcal{M}$. However, for some encoder functions (symmetric or otherwise) messages in the preimage of $\mathcal{C}$ are invalid messages, thus reducing the amount of different valid messages that can be decoded from some of the ciphertexts, and consequently reducing the message equivocation.

This is illustrated by the following simple example. Let $\mathcal{M}_v = \{00, 01, 10\}$, $\mathcal{M}_i = \{11\}$, $\mathcal{K} = \{00, 11\}$, and $enc(m, k) = m \oplus k$, where $\oplus$ represents the exclusive disjunction operator. Then enc is as depicted in Table 2a.

		Message			
		Valid			**Invalid**
		00	**01**	**10**	**11**
Key	**00**	00	01	10	11
	11	11	10	01	00

(a) Encoder function for a 2-bit message and a 1-bit key.

		$P(c)$	$H(M\|C = c)$
Ciphertext c	**00**	1/6	0
	01	1/3	1
	10	1/3	1
	11	1/6	0

(b) Probability and posterior entropy for each ciphertext.

Table 2: XOR encoding example.

Assuming uniform distribution on the key and valid message spaces, max-equivocation is achieved when $H(M|C) = H(K) = 1$ bit. We can compute the equivocation as $H(M|C) = \sum_{c \in C} P(c) H(M|C = c)$. The probability and posterior entropy of each ciphertext is depicted in Table 2b. The resulting message equivocation is $\frac{2}{3}$ bits, which is inferior to the optimal value of 1.

The reason for the sub-optimality of the encoding is that some of the ciphertexts, namely 00 and 11, can map back to invalid messages. If the attacker intercepts the ciphertext 00 they know that the message must be either 00 or 11, and since 11 is invalid this allows them to conclude that the message is necessarily 00. On the other hand, if they intercept the ciphertext 01 they have no way to decide whether the message was 01 or 10.

This example shows that when the preimage of the ciphertext space includes invalid messages the security of the transmission is reduced, and this is captured by the fact that message equivocation decreases accordingly. Now consider the encoder function $enc(m, k) = mod((m+k),\ 3)$ when m is valid and $enc(m, k) = 11$ otherwise, where mod is the standard modulus function, on the same message spaces and $\mathcal{K} = \{0, 1\}$, depicted in Table 3.

		Message			
		Valid			**Invalid**
		00	**01**	**10**	**11**
Key	**0**	00	01	10	11
	1	01	10	00	11

Table 3: Sum-modulo encoding for a 2-bit message and a 1-bit key

In this case each the only ciphertexts that can be produced are 00, 01 and 10; let's call them *valid ciphertexts*. Each valid ciphertext maps back to exactly 2 possible valid messages, so the message equivocation is 1 bit and max-equivocation is achieved. In fact, the symbols in the column of the invalid messages do not matter as long as the valid ciphertexts map to the right valid messages. Even substituting a constant for all the ciphertexts of the invalid messages would not change the equivocation of the system, even though it would violate the semi-injectivity on the key condition of a symmetric encoder function.

In fact, in the next section we will consider encoder functions that are not symmetric, i.e. are not semi-injective on their keys, to show they can sometimes yield improved results over symmetric encoders and generally stronger encryption.

7. Relaxing Encoder Injectivity

The semi-injectivity conditions on the symmetric encoder functions imply that there exist at most one message that can produce each key-ciphertext pair and at most one key that can produce each message-ciphertext pair.

Relaxing the semi-injectivity on the message would mean that the receiver of the ciphertext, knowing the key, would not be able to uniquely determine which message the sender meant to send him. This would not be desirable.

However, if the messages are divided into valid and invalid messages as described in Section 6, it is sufficient for the key-ciphertext pair to map back to a single valid and any number of invalid messages for the receiver to be able to single out the valid message that the attacker sent. For this reason, we can relax the "semi-injectivity on $\mathcal{M}$" condition on encoder functions to "semi-injectivity on $\mathcal{M}_v$".

Similarly, we here exploit the weakening of the definition of a shared-key cryptosystem to exploit the lack of semi-injectivity on the key. This allows that there are ciphertexts that can be decoded as the same message by different keys. Since the receiver has the correct key this would not be a problem, and we can show that in general relaxing this condition can produce stronger encryption. Thus, in this section we will consider encoder functions from Definition 9 instead of symmetric encoder functions from Definition 8.

7.1. Example

To illustrate this, we recall the example in Table 1. Let $\mathcal{M} = \{m_1, m_2, m_3\}$ with distribution $\rho(M) = \{m_1 \mapsto 0.02, m_2 \mapsto 0.49, m_3 \mapsto 0.49\}$ and let the key be uniform on $\mathcal{K} = \{k_1, k_2\}$.

		Message		
		m_1	m_2	m_3
Key	k_1	c_1	c_2	c_3
	k_2	c_2	c_3	c_1

(a) Best Latin rectangle

		Message		
		m_1	m_2	m_3
Key	k_1	c_1	c_2	c_3
	k_2	c_4	c_3	c_2

(b) Optimal symmetric encoder (with semi-injectivity on key)

		Message		
		m_1	m_2	m_3
Key	k_1	c_1	c_2	c_3
	k_2	c_1	c_3	c_2

(c) Optimal asymmetric encoder (without semi-injectivity on key)

Table 4: Encoder functions for message distribution $\rho(M) = \{m_1 \mapsto 0.02, m_2 \mapsto 0.49, m_3 \mapsto 0.49\}$ and uniform key distribution.

The theoretical maximum to the message equivocation is achieved when the entropy of the ciphertext is minimal at $H(C) = H(M) \approx 1.1214$. The best Latin rectangle and best symmetric encoder function depicted in Table 4a and 4b respectively do not achieve this, having a ciphertext entropy of ≈ 1.5097 bits and ≈ 1.1414 respectively.

However, if we do not enforce semi-injectivity on the key, we can have the encoder function in Table 4c. Although this encoder function achieves the same message equivocation as the encoder in Table 4b, they differ in their key equivocation. This can have significant impact in scenarios where the key is re-used.

Consider when the same key $k \in \{k_1, k_2\}$ is used to transmit two messages $\alpha, \beta \in \{m_1, m_2, m_3\}$. If the encoder in Table 4b is used twice then the equivocation is $H(M_\alpha, M_\beta | C_\alpha, C_\beta) = 0.9604$. However, if instead the encoder in

Table 4c is used twice the equivocation is $H(M_\alpha, M_\beta | C_\alpha, C_\beta) = 0.9996$. More generally, when using a single key, the encoder in Table 4b has equivocation $H(\vec{M}|\vec{C}) = 0.98^n$ for transmitting n messages, this is due to any collection of messages containing m_1 completely leaking the key. By contrast, the encoder in Table 4c has equivocation $H(\vec{M}|\vec{C}) = 1 - 0.02^n$ for n messages, due to m_1 not leaking any information about the key.

Asymmetric encoders both generalise, and can achieve better equivocation than, symmetric encoders. Although this is demonstrated above on a key re-use scenario, asymmetric encoders perform better than symmetric encoders in non-trivial single transmission scenarios as well.

7.2. Differences in the Theory

Some of the results of Section 3 depend on the encoder function being symmetric and thus semi-injective on the key. We investigate the impact of using general encoder functions on the lemmata and theorems of Section 3.

Not having semi-injectivity on $\mathcal{K}$ means that the equation $H(K|M, C) = 0$ does not hold anymore. Consequently, Theorem 1 that was based on this result does not hold anymore, meaning that the equivocation of the message $H(M|C)$ and the equivocation of the key $H(K|C)$ can be different.

Lemma 2 still holds, but the lower bound on Lemma 3 was based on using Lemma 1 on both $\mathcal{M}$ and $\mathcal{K}$, so the lower bound of Lemma 3 becomes $H(M) \leq H(C)$.

Lemma 4 does not hold, meaning that $H(C|M)$ is not necessarily $H(K)$ but will depend on the particular encoder function producing $\mathcal{C}$.

As a consequence, the upper and lower bounds for message equivocation in Theorem 2 change to

$$H(C|M) - H(K) \leq H(M|C) \leq H(C|M) \ .$$

Minimizing the entropy of the ciphertext is not sufficient to maximize message equivocation anymore, invalidating Corollary 1. Instead, message equivocation is maximized when the difference $H(C|M) - H(C)$ approaches 0, posing a more complex optimization problem since both elements depend on the encoder function. In fact such difference is the negative of mutual information, since

$$H(C|M) - H(C) = -(H(C) - H(M|C)) = -I(M; C) \ .$$

This confirms that to maximize message equivocation we must minimize the mutual information between message and ciphertext, as we already noted in Section 3.

As for the results in Section 4.2, the upper bound of Theorem 3 becomes $H(K|C) \leq H(K)$, making the theorem uninteresting since $0 \leq H(X|Y) \leq H(X)$ holds for any two random variables X, Y. Finally, Theorem 4 becomes

$$H(M) - H(C|M) \leq I(M, K; C) \leq H(M) - H(C|M) + 2H(K) \ .$$

8. Max-equivocation in Practice

In this section we discuss how an unconditionally secure communication system using the results of this paper can be constructed in practice, including the evaluation of heuristics for the generation of suitable encoder functions.

8.1. Scenario

As usual, `A` wants to send messages to `B` on an insecure channel in a way that `E` cannot understand even by collecting the ciphertexts. Recall that `E` can intercept the communications on the channel but not modify them.

First of all, `A` and `B` need to agree on an encoder function *enc*. The encoder function can be public, so `A` generates *enc* and sends it to `B` on the insecure channel, where `E` intercepts it. Now all three agents know the encoder function *enc*, and consequently the message space $\mathcal{M}$ and key space $\mathcal{K}$. `B` and `E` reverse the encoder function obtaining the corresponding decoder *dec*. Note that the generation, transmission and reversal of the encoder function has to be performed only once.

Before starting the transmission of ciphertexts, `A` and `B` use a key distribution protocol to agree on a given key $k_1 \in \mathcal{K}$. They know that the entropy of `E` over the key is $H(K)$.

Now, `A` can start sending messages to `B`. Whenever `A` wants to send a message m_α to `B`, `A` encodes the message m_α with the key k_1 and sends the corresponding ciphertext c_α to `B` on the insecure channel, where c_α is also intercepted by `E`.

Each subsequent transmission of messages with the same key k_1 reduces the entropy of k_1 and the equivocation of all messages transmitted with k_1. For instance, if `A` sends to `B` three messages m_α, m_β and m_γ encoded as ciphertexts c_α, c_β and c_γ respectively, the remaining entropy of k_1 is $H(K|C_\alpha, C_\beta, C_\gamma)$ and the equivocation of each of the three messages is $H(M|C_\alpha, C_\beta, C_\gamma)$. When this value becomes too low, `A` and `B` run the key distribution protocol again to agree on a new shared key k_2, resetting the key entropy to $H(K)$, and continue the transmission.

Since the generation of the encoder happens only once and potentially well before the transmission starts, `A` could spend significant time in the creation of an optimal encoder. However, finding the optimal encoder may be very expensive. Also, the quality of the encoder depends on the probability distribution of the messages that `A` will send, and `A` may have only an idea of such distribution, making perfect optimization pointless.

Therefore, instead of focusing on the optimal encoder for a given message distribution, we will consider more practical heuristic algorithms to produce a "good enough" encoder in a reasonable time, and we will experiment with the heuristics to find out which one can produce the best encoder in the least time.

8.2. Symmetric Encoder Generation Heuristics

We consider the generation of symmetric encoder functions via efficient heuristics. Our heuristics generate symmetric encoders by greedy sequential approximation. The quality of symmetric encoders is easier to compare than the quality of asymmetric encoders, since in the former case it is sufficient to choose the symmetric encoder that produces the ciphertext with the lowest entropy. While as explained in Section 7 symmetric encoders are non-optimal in general, the equivocation obtained by symmetric encoders is close enough to the (unachievable) theoretical maximum. Symmetric encoders are easier to evaluate, and thus more usable in practice. For these reasons, in this section we focus on symmetric encoders.

We propose 4 different heuristic algorithms that produce a symmetric encoder function for a given message space $\mathcal{M} = \{m_1, m_2, \ldots, m_{|\mathcal{M}|}\}$ with a given probability distribution $\rho(M)$ and a given key space $\mathcal{K} = \{k_1, k_2, \ldots, k_{|\mathcal{K}|}\}$ with uniform distribution. We will assume that $|\mathcal{K}| < |\mathcal{M}|$ and $\rho(m_1) \leq \rho(m_2) \leq \cdots \leq \rho(m_{|\mathcal{M}|})$.

The algorithms we propose are similar as they all consider the symmetric encoder function as a $|\mathcal{K}| \times |\mathcal{M}|$ matrix $\mathbf{C}$ where the rows correspond to values of the key and the columns to the values of the message, as shown in the tables in this paper. The cell $\mathbf{C}_{i,j}$ contains the ciphertext symbol produced when the key is k_i and message is m_j. For brevity we denote by $H(\mathbf{C})$ the entropy of the ciphertext induced by the symmetric encoder function represented by $\mathbf{C}$.

The algorithms initialize $\mathbf{C}$ by assigning to each value in the column m_j the symbol c_j, for $1 \leq j \leq |\mathcal{M}|$. The resulting ciphertext would have exactly $H(C) = H(M)$ and thus achieve max-equivocation, however it is not a valid symmetric encoder function since the same symbol is repeated more than once in each column. The algorithms then transform $\mathbf{C}$ into a valid symmetric encoder function while greedily trying to increase the entropy of the ciphertext by the smallest possible amount.

For $\mathbf{C}$ to be a symmetric encoder function, it must be that in each row and column of $\mathbf{C}$ each symbol appears at most once. We call a symbol $c = \mathbf{C}_{i,j}$ a *conflict* if it violates this property, i.e. if c appears more than once in row i or column j.

The algorithms proceed by reducing the number of conflicts in $\mathbf{C}$, and returning $\mathbf{C}$ when there are no conflicts left and consequently $\mathbf{C}$ represents a symmetric encoder function. A conflict can be replaced in two different ways: by *swapping* it with a symbol in a different position in $\mathbf{C}$ or by *transforming* it to another symbol.

Swapping $c = \mathbf{C}_{i,j}$ and $c' = \mathbf{C}_{i',j'}$ means that we simultaneously set $\mathbf{C}_{i,j} \leftarrow c'$ and $\mathbf{C}_{i',j'} \leftarrow c$. A swap is considered valid only if c is not a conflict in position (i', j') and c' is not a conflict in position (i, j). The algorithms perform only valid swaps, so each swap strictly reduces the number of conflicts in $\mathbf{C}$. We will denote the result of the swap with $\mathbf{C}[c \leftrightarrow c']$. A *best swap* operation $\overline{\mathbf{C}}[c \leftrightarrow c']$ for a given conflict c is a valid swap such that $\forall c'' \in \mathbf{C}.\ H(\overline{\mathbf{C}}[c \leftrightarrow c']) \leq H(\mathbf{C}[c \leftrightarrow c''])$.

Transforming $c = \mathbf{C}_{i,j}$ means finding a symbol c_ν that does not appear in row i nor in column j and setting $\mathbf{C}_{i,j} \leftarrow c_\nu$. A transformation reduces the number of conflicts by one. We will denote the result of the transformation with $\mathbf{C}[c \leftarrow c_\nu]$. Note that no symmetric encoder function has more than $|\mathcal{M} \times \mathcal{K}|$ different symbols. A *best transformation* operation $\overline{\mathbf{C}}[c \leftarrow c_\nu]$ for a given conflict c is a transformation such that $\forall c' \in \{c_1, c_2, ..., c_{|\mathcal{M}\times\mathcal{K}|}\}.\ H(\overline{\mathbf{C}}[c \leftarrow$

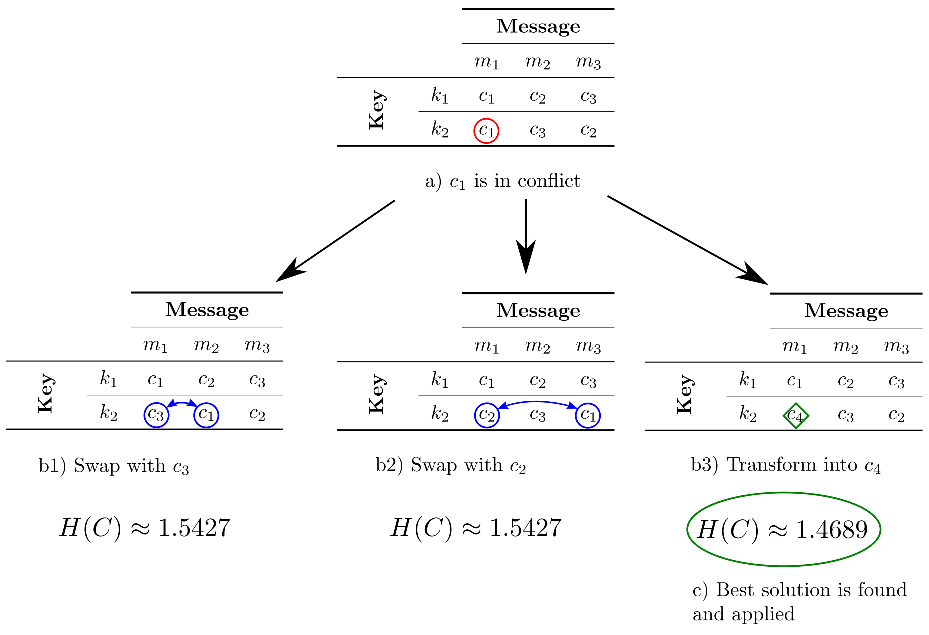

Figure 3: Conflict resolution example for an encoder with message space $\mathcal{M} = \{m_1, m_2, m_3\}$ with distribution $\rho(M) = \{m_1 \mapsto 0.1, m_2 \mapsto 0.45, m_3 \mapsto 0.45\}$ and key uniformly distributed on $\mathcal{K} = \{k_1, k_2\}$. Ciphertext c_1 in the first column is in conflict (a). The possible options to resolve the conflict are considered: swapping c_1 with c_2 (b1), swapping c_1 with c_2 (b2), or transforming c_1 into a new ciphertext symbol c_4 (b3). The ciphertext entropy of each option is computed. The transformation is found to have the lowest ciphertext entropy (c), so it is chosen and applied.

$c_\nu]) \leq H(\mathbf{C}[c \leftarrow c'])$.

In Figure 3 we show an example of conflict resolution for an encoder with message space $\mathcal{M} = \{m_1, m_2, m_3\}$ with distribution $\rho(M) = \{m_1 \mapsto 0.1, m_2 \mapsto 0.45, m_3 \mapsto 0.45\}$ and key uniformly distributed on $\mathcal{K} = \{k_1, k_2\}$.

All algorithms always terminate. We introduce them and explain how they differ.

The IncRow algorithm scans $\mathbf{C}$'s columns in increasing order of probability, and each column from top to bottom. When it finds a conflict c, it considers the best swap $\overline{\mathbf{C}}[c \leftrightarrow c']$ of c with elements on the same row and the best transformation of c with a different symbol $\overline{\mathbf{C}}[c \leftarrow c_\nu]$. If $H(\overline{\mathbf{C}}[c \leftarrow c_\nu]) < H(\overline{\mathbf{C}}[c \leftrightarrow c'])$ it performs the best transformation, otherwise the best swap. The pseudocode of IncRow is described in Algorithm 1.

The DecRow algorithm is equivalent to IncRow except that the columns are scanned in decreasing order of probability. The pseudocode of DecRow is the same as the one described in Algorithm 1 except that the order of the **for** on line 2 is $j \leftarrow |\mathcal{M}|$ *to* 1.

The RandRow algorithm is equivalent to IncRow except that the columns are scanned in a random order. The pseudocode of RandRow is the same as

```
Data: message space $\mathcal{M} = \{m_1, m_2, \ldots, m_{|\mathcal{M}|}\}$, probability distribution
      $\rho(M)$, key space $\mathcal{K} = \{k_1, k_2, \ldots, k_{|\mathcal{K}|}\}$.
Result: matrix form $\mathbf{C}$ of a symmetric encoder enc.
1  Set $\mathbf{C}_{i,j} = c_j$, $\forall 1 \leq i \leq |\mathcal{K}|, 1 \leq j \leq |\mathcal{M}|$;
2  for $j \leftarrow 1$ to $|\mathcal{M}|$ do
3      for $i \leftarrow 1$ to $|\mathcal{K}|$ do
4          if $c = \mathbf{C}_{i,j}$ is a conflict then
5              Find the best swap $\overline{\mathbf{C}}[c \leftrightarrow c']$ with any element $c'$ on row $i$;
6              Find the best transformation $\overline{\mathbf{C}}[c \leftarrow c_\nu]$;
7              if $H(\overline{\mathbf{C}}[c \leftarrow c_\nu]) < H(\overline{\mathbf{C}}[c \leftrightarrow c'])$ then
8                  $\mathbf{C} \leftarrow \overline{\mathbf{C}}[c \leftarrow c_\nu]$
9              else
10                 $\mathbf{C} \leftarrow \overline{\mathbf{C}}[c \leftrightarrow c'])$
11             end
12         end
13     end
14 end
15 Return $\mathbf{C}$;
```

Algorithm 1: IncRow

the one described in Algorithm 1 except that the order of the columns of $\mathbf{C}$ is randomized after line 1.

The Extensive algorithm considers all conflicts in $\mathbf{C}$ and for each one the best swap and the best transformation, and then it performs the swap or transformation operation that is the best among all possible conflict resolutions (ties in favor of swaps). It repeats this step until there are no conflicts left. The pseudocode of Extensive is described in Algorithm 2.

Lemma 7. *Algorithms* IncRow*,* DecRow*,* RandRow *and* Extensive *terminate.*

Proof. Since each iteration of the main cycle of each algorithm reduces the number of conflicts. □

We now perform experiments on the proposed heuristics to determine which one is the most effective in producing a symmetric encoder function approaching max-equivocation.

8.3. Experimental Methodology

The experimental results are from two different phases to explore the behaviour and performance of different algorithms. The first phase considers a variety of small message spaces with two different message distribution generation algorithms (Split and Skew). The goal of this phase is to identify which algorithms perform comparatively well. The second phase iterates over increasing size of the message spaces with a single message distribution generation

```
Data: message space $\mathcal{M} = \{m_1, m_2, \ldots, m_{|\mathcal{M}|}\}$, probability distribution
      $\rho(M)$, key space $\mathcal{K} = \{k_1, k_2, \ldots, k_{|\mathcal{K}|}\}$.
Result: matrix form $\mathbf{C}$ of a symmetric encoder $enc$.
1  Set $\mathbf{C}_{i,j} = c_j, \forall 1 \leq i \leq |\mathcal{K}|, 1 \leq j \leq |\mathcal{M}|$;
2  while $\mathbf{C}$ is not a symmetric encoder function do
3      Let $min\mathbf{C} \leftarrow \mathbf{C}$, $minH \leftarrow \infty$;
4      foreach conflict $c$ do
5          Find the best swap $\overline{\mathbf{C}}[c \leftrightarrow c']$ with any element $c' \in \mathbf{C}$;
6          Find the best transformation $\overline{\mathbf{C}}[c \leftarrow c_\nu]$;
7          if $H(\overline{\mathbf{C}}[c \leftarrow c_\nu]) < H(\overline{\mathbf{C}}[c \leftrightarrow c'])$ then
8              Let $candidate\mathbf{C} \leftarrow \overline{\mathbf{C}}[c \leftarrow c_\nu]$;
9          else
10             Let $candidate\mathbf{C} \leftarrow \overline{\mathbf{C}}[c \leftrightarrow c']$;
11         end
12         if $H(candidate\mathbf{C}) < minH$ then
13             $min\mathbf{C} \leftarrow candidate\mathbf{C}$;
14             $minH \leftarrow H(candidate\mathbf{C})$;
15     end
16     $\mathbf{C} \leftarrow min\mathbf{C}$;
17 end
18 Return $\mathbf{C}$;
```

Algorithm 2: EXTENSIVE

algorithm to test the performance of refined encoder generators. In both phases the key space size is randomly chosen to be between 2 and the maximum $|K|$ such that $H(K) < H(M)$.

The SPLIT algorithm splits the probability available x (initially 1) randomly into two parts x_1 and x_2, and then recursively calls itself with these parts of the two halves of the message (sub-)space. This continues until the subspace is a single message that is assigned the whole probability.

The SKEW algorithm splits the probability available into two parts, assigns the first part to the current message (initially the first), and then continues with the remaining probability and remaining messages. This tends to generate more skewed probability distributions than SPLIT.

In both phases, once the size of the message space, message distribution, and size of the key space have been chosen, each algorithm is run in turn on the same initial state, with the data recorded after each symmetric encoder function has been created.

The code to perform these tests is written in Java 1.8 and run on Linux 3.13 64-bit kernel on an Intel Core i7-3720QM 2.60GHz CPU with 8GB of RAM. The code used to run these experiments is available upon request.

8.4. Comparing Algorithms

This phase generates tests with random small-sized message spaces, message distributions, and key sizes; determining the domain for an encryption function. The different algorithms for generating encryption functions are then all run on the same random examples and their outputs compared, in particular:

Win% The percentage of times each algorithm generates the lowest entropy (may sum to over 100% since more than one algorithm can generate the same lowest entropy).

Δ The average percentage of the message entropy that is lost; that is the average $\frac{(H(C)-H(M))\times 100}{H(M)}$.

Runtime The average runtime for the algorithm in milliseconds.

The results for comparing IncRow, DecRow, RandRow, and Extensive over 450,000 tests (equally split across the two probability generation algorithms) are shown in Table 5. The results indicate that the DecRow algorithm

Algorithm	**Win%**	Δ	**Runtime**
IncRow	9.3462	11.04	0.3978
DecRow	80.5007	9.00	0.2909
RandRow	21.5667	10.31	0.3608
Extensive	8.4744	11.00	172.4039

Table 5: Comparison of IncRow, DecRow, RandRow, and Extensive

performs the best in all categories, followed by RandRow, then IncRow and finally Extensive. In particular the results show that Extensive performs by far the worst, with much higher runtimes yielding the lowest win percentage and only negligibly improved loss of message entropy over the next worse IncRow. Although IncRow is not significantly slower, the low win percentage and highest entropy loss suggest that despite greediness not being optimal, beginning with the smallest probabilities is worse than a random selection. Although DecRow is not optimal, it does prove to be the most effective of the algorithms here and provides a good basis for further experiments.

8.5. Refinement

In the second phase, we consider refinements to the DecRow algorithm that improve the runtime with only small losses in Δ. There are three different approaches used: splitting the symmetric encoder function into subsections by size; splitting the symmetric encoder function into subsections by some probability metric; and considering a rolling aperture instead of the whole symmetric encoder function space.

Splitting into subsections by size is a simple approach that can yield reasonably large improvements. The most trivial algorithm is DualDec that splits

Message			Δ		
size	**DecRow**	**2KSplit**	**Split25**	**Aperture3**	**Drop25**
2^5	.2423769	.2455512	.2460305	.2449590	.2923967
2^6	.2827137	.2881369	.2902109	.2857851	.3233435
2^7	.3221931	.3305574	.3321461	.3258651	.3550239
2^8	.3588024	.3692356	.3695319	.3630879	.3856449
2^9	.3971868	.4095588	.4081074	.4019161	.4198576

Table 6: Comparison of Δ of DecRow, 2KSplit, Split25, Aperture3, and Drop25 over different message sizes. Message size is in bits, smaller Δ is better.

Message			**Runtime**		
size	**DecRow**	**2KSplit**	**Split25**	**Aperture3**	**Drop25**
2^5	.46611	.32542	.23656	.26828	.26374
2^6	3.6288	2.4475	1.8236	2.0277	2.1740
2^7	30.589	19.768	16.501	17.228	19.903
2^8	287.16	182.27	163.63	164.65	197.87
2^9	3427.6	2086.6	1983.4	1981.3	2336.8

Table 7: Comparison of runtime of DecRow, 2KSplit, Split25, Aperture3, and Drop25 over different message sizes. Message size is in bits, runtime in milliseconds, smaller runtime is better.

the symmetric encoder function space in half and runs DecRow on each half. Another straightforward approach is to split into equal subsections of a given size. This was tested with subsections equal to the key space KSplit and subsections twice the size of the key space 2KSplit.

A further refinement of the splitting into subsections approach was to allow overlap between the subsections. This was implemented as the family of SplitXX algorithms that used different amounts of overlap between the subsections. The results here are detailed for the Split25 algorithm that extended 25% of the key space outside each subsection (or 3 messages for key sizes less than 12).

Rather than splitting the symmetric encoder function into sections with possible overlap, the aperture approach simply limits how far above and below the current message to consider swapping. Since the messages are ordered on probability, it was hypothesised that a swap would not be more than 3 times the key space below the current message, and only a few messages above (3). This is represented in the results by the Aperture3 algorithm.

Inspired by the necessary and sufficient conditions for max-equivocation in Theorem 6 an alternative approach was to try and find a drop point in the message distribution where the difference in probability between two messages is greater than a running average probability. This attempts to split the symmetric encoder function into sections of least key space size with more uniform

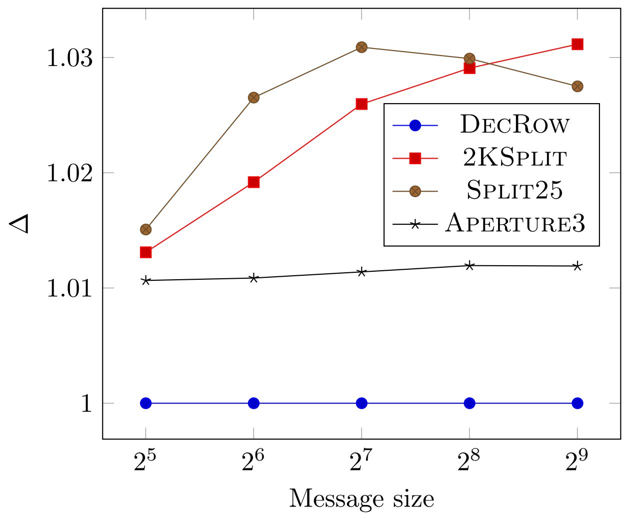

Figure 4: Comparison of Δ's for DecRow, 2KSplit, Split25, and Aperture3 versus message size

probabilities within each section. The drop point was taken to be a 25% different to the current average in the Drop25 algorithm.

The following results consider the DecRow, 2KSplit, Split25, Aperture3, and Drop25 algorithms run while comparing message sizes. (Other algorithms and variations were tested, but performed worse than those chosen and so have been omitted from detailed exploration here.) Tables 6 & 7 shows the results for 100,000 tests with fixed message size (and random key space size such that $1 \leq H(K) < H(M)$). Message probability distributions are generated by the Split probability generation algorithm since Skew tends to be too skewed for larger message sizes. The deltas relative to DecRow are shown in Figure 4, while the runtimes relative to DecRow are shown in Figure 5.

The data shows that the Aperture3 algorithm consistently produces the symmetric encoder with the delta closer to DecRow.

The data shows that the Aperture3 algorithm consistently produces symmetric encoders with delta around 1% worse than DecRow, while doing so in less than 60% of the runtime. The Split25 algorithm performs worse for small message sizes but appears to improve as the message size increases, while having the best runtimes. 2KSplit appears to be increasingly worse with respect to delta although the runtime improvements increase with message size. Lastly, Drop25 was not graphed in Figure 4 due to performing significantly worse, and the runtime does not appear to be an improvement over other refinements.

9. Conclusions

We have presented max-equivocation, a generalization of Shannon's perfect secrecy condition that can also be established when the entropy of the key

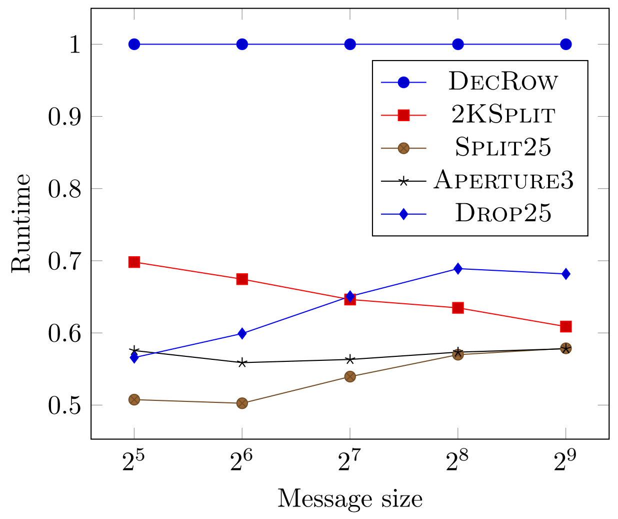

Figure 5: Comparison of runtimes for DecRow, 2KSplit, Split25, Aperture3, and Drop25 versus message size

is smaller of the entropy of the message. Message equivocation measures the number of different messages that can be decrypted from the ciphertext, and max-equivocation holds when this number is maximized. Max-equivocation is obtained when mutual information is minimized, and for symmetric encoder functions this corresponds to minimizing the entropy of the ciphertext.

We have shown that max-equivocation can be easily applied to scenarios with key re-use. Also that having keys with more entropy than the messages immediately loses the excess entropy, performing very poorly from a key re-use perspective.

We have shown that having a ciphertext space larger than the message space can increase the effectiveness of encoder functions, proving the general non-optimality of encoder functions that correspond to Latin rectangles. We have shown that in general it is impossible to give a symmetric encoder function that achieves max-equivocation: max-equivocation is achievable if and only if the message space can be divided in subspaces of size at least $|\mathcal{K}|$ each of which has uniform distribution.

We have addressed the unicity problem by considering messages that could be decoded from ciphertexts but have probability 0 of being chosen by the sender. We show that this can be resolved by either relaxing semi-injectivity on the key, or by distinguishing valid and invalid messages.

We considered the consequences of relaxing the semi-injectivity on the key encoder condition. The consequent lack of symmetry in the behavior of the encoder towards messages and keys complicates some of the results in Section 3. This allows us to produce more effective encoders, at significant complexity cost.

We have compared four heuristics to efficiently derive an effective encoder function. We established that the DecRow algorithm works better than the others under our experimental parameters, consistently providing encoder functions that are only a few percentage points worst than the theoretical optimum and in less time than the other heuristics.

We show that refinements to DecRow can yield algorithms that perform almost as well with respect to entropy, while doing so in almost half the runtime.

[1] Alvim, M. S., Chatzikokolakis, K., Palamidessi, C., Smith, G., 2012. Measuring information leakage using generalized gain functions. In: Chong, S. (Ed.), 25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012. IEEE, pp. 265–279.

[2] Biondi, F., Given-Wilson, T., Legay, A., 2015. Attainable unconditional security for shared-key cryptosystems. In: 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015, Helsinki, Finland, August 20-22, 2015. IEEE, pp. 1–8.

[3] Bruen, A., Forcinito, M., 2011. Cryptography, Information Theory, and Error-Correction: A Handbook for the 21st Century. Wiley.

[4] Csiszár, I., Körner, J., 1978. Broadcast channels with confidential messages. IEEE Transactions on Information Theory 24 (3), 339–348.

[5] Ho, S., Chan, T., Uduwerelle, C., 2011. Error-free perfect-secrecy systems. In: Kuleshov, A., Blinovsky, V., Ephremides, A. (Eds.), 2011 IEEE International Symposium on Information Theory Proceedings, ISIT 2011, St. Petersburg, Russia, July 31 - August 5, 2011. IEEE, pp. 1613–1617.

[6] Köpf, B., Basin, D. A., 2011. Automatically deriving information-theoretic bounds for adaptive side-channel attacks. Journal of Computer Security 19 (1), 1–31.

[7] Mileva, A., Markovski, S., 2013. Quasigroup representation of some Feistel and generalized Feistel ciphers. In: ICT Innovations 2012. Vol. 207. pp. 161–171.

[8] Russell, A., Wang, H., 2006. How to fool an unbounded adversary with a short key. IEEE Transactions on Information Theory 52 (3), 1130–1140.

[9] Shannon, C. E., Jul. 1948. A mathematical theory of communication. The Bell system technical journal 27, 379–423.

[10] Shannon, C. E., 1949. Communication Theory of Secrecy Systems. Bell System Technical Journal 28.

[11] Smith, G., 2009. On the foundations of quantitative information flow. In: de Alfaro, L. (Ed.), Foundations of Software Science and Computational Structures, 12th International Conference, FOSSACS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings. Vol. 5504 of Lecture Notes in Computer Science. Springer, pp. 288–302.

[12] Welsh, D., 1988. Codes and Cryptography. Clarendon Press, New York, NY, USA.

[13] Wu, Y., Noonan, J. P., Agaian, S. S., 2011. Dynamic and implicit latin square doubly stochastic s-boxes with reversibility. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Anchorage, Alaska, USA, October 9-12, 2011. IEEE, pp. 3358–3364.

[14] Wu, Y., Zhou, Y., Noonan, J. P., Agaian, S. S., 2014. Design of image cipher using Latin squares. Inf. Sci. 264, 317–339.

Fabrizio Biondi obtained his BSc and MSc in Computer Science from the University of Bologna, Italy. He then undertook a PhD with Andrzej Wąsowski at the IT University of Copenhagen, developing techniques for the computation of information leakage of systems modeled as Markovian processes. Since his graduation in 2014 he has been working for the Inria French national institute for Computer Science in Rennes in the team of Axel Legay. Fabrizio's research is focused on applications of probability and information theory to computer security, information leakage computation, unconditional cryptography, malware detection and fingerprinting, and static analysis of white-box systems.

Thomas Given-Wilson holds a BCST from the University of Sydney, and both BS(Hons)IT and PhD from the University of Technology, Sydney. He worked on static analysis tools for NICTA before moving to France to join Inria as a post-doctoral researcher. Thomas' research is focused upon privacy and security, foundations of computation and concurrency, and quantified information flow.

Axel Legay held positions at University of Liège and CMU (under the supervision of Ed Clarke). He is a full-time researcher at INRIA where he leads research on the Internet of Things and systems of systems. His main research interests are in developing formal specification and verification techniques for Software Engineering. Axel is a referee for top journals and conferences in formal verification and simulation, and program co-chair of INFINITY'09, FIT'10, Runtime Verification 2013, Splat 2014, FORMATS 2014, ATVA 2016, and TACAS 2017. He has been the Inria representative for more than 10 Inria projects over the six last years.