Authenticated Key Distribution: When the Coupon Collector is Your Enemy

Marc Beunardeau[1], Fatima-Ezzahra El Orche[2,4], Diana Maimuţ[3], David Naccache[2,4], Peter B. Rønne[4], and Peter Y.A. Ryan[4]

[1] Nomadic Labs, Paris, France
marc.beunardeau@nomadic-labs.com
[2] ENS, CNRS, PSL Research University, Paris, France
{fatimaezzahra.elorche,david.naccache}@ens.fr
[3] Advanced Technologies Institute, Bucharest, Romania
diana.maimut@dcti.ro
[4] SnT, FSTC, University of Luxembourg
{fatima.elorche,peter.roenne,peter.ryan}@uni.lu

Abstract We introduce new authenticated key exchange protocols which on the one hand do not resort to standard public key setups with corresponding assumptions of computationally hard problems, but on the other hand, are more efficient than distributing symmetric keys among the participants. To this end, we rely on a trusted central authority distributing key material whose size is independent of the total number of users, and which allows the users to obtain shared secret keys. We analyze the security of our construction, taking into account various attack models. Importantly, only symmetric primitives are needed in the protocol making it an alternative to quantum-safe key exchange protocols which rely on hardness assumptions.

Keywords: Symmetric cryptography, key exchange protocol, authentication, provable security, post-quantum cryptography.

1 Introduction

Symmetric key primitives are the preferred choice for fast encryption applications. On the other hand, public-key cryptography is widely adopted for ensuring (authenticated) key exchange functionalities. Many currently deployed applications take the best of both worlds and use key encapsulation mechanisms where keys are exchanged using public key protocols and are subsequently used as input to efficient symmetric primitives.

This paper proposes an intermediate construction. We introduce a cryptographic protocol approaching some of the functionalities of public-key encryption while relying entirely on *symmetric* primitives. Before we proceed, we stress that our models are very different from those of classical public-key cryptography, and so are their security and efficiency metrics. However, it appears that *in many* *practical settings*, the proposed constructions can successfully replace classical public-key encryption.

Given that our techniques do not resort to number-theoretic cryptography, the construction is naturally resistant against attacks from quantum computers.

Prior Work. Key exchange protocols play an important role in protecting end-to-end communications. Initially introduced in [5], the previously mentioned notion revolutionized cryptology. These protocols allow two parties to generate securely a common secret key, which will be used later for different cryptographic purposes such as sending authenticated and encrypted messages. Another closely related flavour of such protocols may be defined as authenticated key exchange protocols. The first basic understandings of this category of schemes were presented in [2,4]. Considering that such constructions could lead to practical and efficient protocols, the authors focused on formalizing the security notions related to entity authentication and key distribution.

Note that contrary to the Needham-Schroeder symmetric key protocol [8], the central authority is only active in the enrolment phase in our protocol, not during the actual key establishment.

ID-based secret key cryptography was first presented in [7]. While the paradigm similarity between this paper and [7] is obvious (*i.e.* mimicking public-key cryptography with symmetric primitives), the technical details are of different nature and granularity. We stress that even though [7] introduces applications like a challenge-response authentication protocol and an ID-based MAC algorithm, it does not provide an in-depth security analysis. Moreover, our key exchange protocol can use more than one key per user, which, as we will see, allows us to optimise security non-trivially.

Structure of the Paper. We present our authenticated key distribution protocol in Section 2, describing particular and general cases. In Section 3, we discuss the adversarial advantage in various attack scenarios, computing probabilities and expectation values. We provide a security analysis of our scheme in Section 4. Finally, we conclude in Section 5 and discuss future work ideas. We introduce notations, definitions and security assumptions used throughout the paper in Appendix A. Appendix B presents the proofs of the lemmas from Section 3. Appendix C tackles parameter choices and discusses the efficiency of our protocol.

2 The Protocol

Participants. Let n be the number of the users in the system (n can be very large, for instance, a billion), each having a unique identity ID_i, where $i \in [1, n]$. In the following, ID_i will designate both the (alphanumeric) name of user i and the user itself as a physical entity. The proposed protocol relies on a central authority (CA) which creates r key tables (called 'racks') each containing ℓ random κ-bit keys. CA distributes, to each user, u distinct keys chosen randomly from each rack, *i.e.* $u \times r$ keys per user. CA also provides each user with supplementary key material that will be described later.

Building-Blocks. Let $f(k, m)$ be a MAC function, where k is the key and m is the message. The protocol also uses a hash function h.

For the sake of clarity, we describe the protocol in steps. We first consider and analyze a basic one-rack case ($r = 1$) and one key per user ($u = 1$).

2.1 Basic Scheme ($r = 1$ and $u = 1$)

Key Generation. CA generates one rack of ℓ secret keys: $\{k_1, \ldots, k_\ell\}$.

User Enrolment. CA then gives to ID_i:

– A secret key $k_{I(i)}$, where $I(i) \in_R [1, \ell]$;
– A table T_i containing the ℓ derived keys: $T_i = \{t_{i,1}, \ldots, t_{i,\ell}\}$ where $t_{i,j} = f(k_{I(j)}, \mathrm{ID}_i)$.

Remark 1. Two users, ID_i and ID_j may get (and in reality are actually expected to get) from CA the same $k_{I(i)} = k_{I(j)}$. Note however that $T_i \neq T_j$ as key tables are derived from identities.

Key Exchange. Assume now that users i and j want to establish a secure communication channel (Figure 2). They proceed as follows:

1. Exchange $I(i)$ and $I(j)$;
2. User j generates $t_{i,I(j)} = f(k_{I(j)}, \mathrm{ID}_i)$;
3. User i generates $t_{j,I(i)} = f(k_{I(i)}, \mathrm{ID}_j)$;
4. Both users generate the common key $\mathsf{sk} = h(t_{i,I(j)}, t_{j,I(i)})$ and use sk to protect their communications.

Remark 2. To avoid ambiguities in the order of parameters of h, we assume that $\mathrm{ID}_i > \mathrm{ID}_j$.

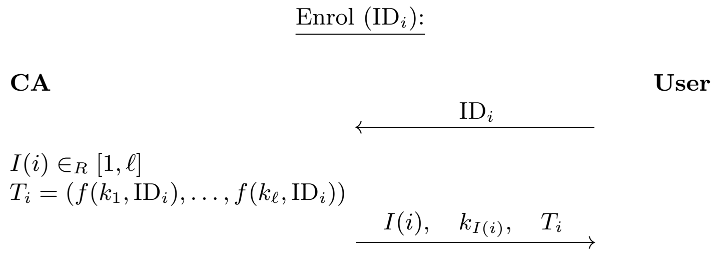
Enrol (ID$_i$):
CA
User
ID$_i$
$I(i) \in_R [1, \ell]$
$T_i = (f(k_1, \mathrm{ID}_i), \ldots, f(k_\ell, \mathrm{ID}_i))$
$I(i)$, $k_{I(i)}$, T_i

Figure 1. User enrolment

ID_i knows $I(i), k_{I(i)}$ and T_i $\qquad\qquad$ ID_j knows $I(j), k_{I(j)}$ and T_j

$$\xrightarrow{I(i)}$$

$$\xleftarrow{I(j)}$$

$t_{j,I(i)} \leftarrow f(k_{I(i)}, \mathrm{ID}_j)$
Read $t_{i,I(j)} = f(k_{I(j)}, \mathrm{ID}_i)$ from T_i
$\mathsf{sk} \leftarrow h(t_{i,I(j)}, t_{j,I(i)})$

$t_{i,I(j)} \leftarrow f(k_{I(j)}, \mathrm{ID}_i)$
Read $t_{j,I(i)} = f(k_{I(i)}, \mathrm{ID}_j)$ from T_j
$\mathsf{sk} \leftarrow h(t_{i,I(j)}, t_{j,I(i)})$

Figure 2. Key exchange

Informally, here is the intuition behind this protocol: we first note that to gain the capacity to listen in to *all* communications, an opponent would need to set his hands on *all* the k_is; this assumes compromising at least ℓ *chosen* devices. Indeed, if at least $t_{i,I(j)}$ or $t_{j,I(i)}$ is unknown, sk is still safe. Evidently, this is not as satisfactory as classical public-key cryptography. Nonetheless, the achieved protection is still useful in many practical scenarios where choosing the target ID_i is impossible[5]. The number of compromised devices required for learning all the ℓ keys with a given probability p is known as the coupon collector's problem (cf. *infra*).

The coupon collector's problem is a famous question introduced in graduate probability lectures. If each box of cookies contains a coupon, and there are ℓ different coupons, what is the probability that more than t boxes need to be bought to collect all ℓ coupons? An alternative statement is: Given ℓ coupons, how many coupons do you expect you need to draw with a replacement before having drawn each coupon at least once? The mathematical analysis of the problem reveals that the expected number of trials needed grows as

$$\ell \log(\ell) + \gamma\ell + \frac{1}{2} + O(\frac{1}{\ell}) \quad \text{where } \gamma = 0.57721\ldots$$

For example, when $\ell = 50$, it takes about 225 trials on average to collect all 50 coupons. We hence see that the defender enjoys *a little advantage* over the attacker. Can this advantage be amplified by engaging in several draws? This is the goal of the next sections.

2.2 General Case: $r \geq 1$ and $u \geq 1$

In this scenario, each user gets u distinct keys per rack. The function I is hence generalized by taking three indices: ① i denoting the concerned user, ② ρ denoting the rack and ③ μ an index running from 1 to u.

In other words, $k^{\rho}_{I(i,\mu,\rho)}$ denotes that the μ-th key from rack ρ is given to user i. Note that $k_{I(i)}$ defined in the previous section just corresponds to $k^1_{I(i,1,1)}$.

[5] For instance, if the ID_is are identity cards, the attacker needs to collect and compromise enough cards hoping to complete his collection of k_is.

Key Generation: CA generates r racks of ℓ distinct keys: $R_\rho = \{k_1^\rho, \ldots, k_\ell^\rho\}$, where $\rho \in [1, r]$.

User Enrolment. CA gives to user ID_i:

– $u \times r$ secret keys:

$$\begin{array}{cccc} k^1_{I(i,1,1)} & k^2_{I(i,1,2)} & \cdots & k^r_{I(i,1,r)} \\ k^1_{I(i,2,1)} & k^2_{I(i,2,2)} & \cdots & k^r_{I(i,2,r)} \\ \vdots & \vdots & & \vdots \\ k^1_{I(i,u,1)} & k^2_{I(i,u,2)} & & k^r_{I(i,u,r)} \end{array}$$

where $\forall \rho \in [1, r], \forall \mu \in [1, u], I(i, \mu, \rho) \in_R [1, \ell]$

– A table T_i of $\ell \times r$ derived keys:

$$T_i = \begin{bmatrix} t^1_{i,1} & t^2_{i,1} & \cdots & t^r_{i,1} \\ t^1_{i,2} & t^2_{i,2} & \cdots & t^r_{i,2} \\ \vdots & \vdots & & \vdots \\ t^1_{i,\ell} & t^2_{i,\ell} & \cdots & t^r_{i,\ell} \end{bmatrix}$$

where $\forall \rho \in [1, r], \forall j \in [1, \ell], t^\rho_{i,j} = f(k^\rho_j, \mathrm{ID}_i)$

Remark 3. Note that the user can derive the table values for his own keys and in principle does not need to store these. In this way memory can be saved at the cost of computational efficiency during key derivation.

Key Exchange: Assume now that users ID_i and ID_j want to establish a secure communication channel. To generate their common secret key, they do the following:

1. Exchange their indices $I(i, \mu, \rho)$ and $I(j, \mu, \rho)$ for $\mu \in [1, u], \rho \in [1, r]$;
2. User ID_i:
   – generates $u \times r$ derived keys:

   $$t^\rho_{j,I(i,\mu,\rho)} = f(k^\rho_{I(i,\mu,\rho)}, \mathrm{ID}_j), \quad \forall \mu \in [1, u], \quad \forall \rho \in [1, r]$$

   – reads $u \times r$ derived keys from his table T_i:

   $$t^\rho_{i,I(j,\mu,\rho)} = f(k^\rho_{I(j,\mu,\rho)}, \mathrm{ID}_i), \quad \forall \mu \in [1, u], \quad \forall \rho \in [1, r]$$

3. User ID_j:
   – generates $u \times r$ derived keys:

   $$t^\rho_{i,I(j,\mu,\rho)} = f(k^\rho_{I(j,\mu,\rho)}, \mathrm{ID}_i) \quad \forall \mu \in [1, u], \quad \forall \rho \in [1, r]$$

– reads $u \times r$ derived keys from his table T_j:

$$t^{\rho}_{j,I(i,\mu,\rho)} = f(k^{\rho}_{I(i,\mu,\rho)}, \mathrm{ID}_j) \quad \forall \mu \in [1,u], \quad \forall \rho \in [1,r]$$

4. Both users ID_i and ID_j generate a common session keys by using h to combine the $2u \times r$ derived keys:

$$\mathsf{sk} = h\big(t^{\rho}_{i,I(j,1,1)}, \dots, t^{\rho}_{i,I(j,u,r)}, t^{\rho}_{j,I(i,1,1)}, \dots, t^{\rho}_{j,I(i,u,r)}\big).$$

<u>Enrol (ID_i):</u>

CA **User**

$$\xleftarrow{\quad \mathrm{ID}_i \quad}$$

$I(i,\mu,\rho) \in_R [1,\ell]$
$\mu \in [1,u], \rho \in [1,r]$
$T_i = (f(k^{\rho}_{I(i,\mu,\rho)}, \mathrm{ID}_i))_{\substack{\mu\in[1,u]\\ \rho\in[1,r]}}$

$$\xrightarrow[\mu \in [1,u], \rho \in [1,r]]{I(i,\mu,\rho), \quad k^{\rho}_{I(i,\mu,\rho)}, \quad T_i}$$

Figure 3. User enrolment for the General Case: $u > 1$ and $r > 1$

Remark 4. For clarity, in Figure 4, we reduce the writing of sk and we write $\mathsf{sk} = h\big(t^{\rho}_{i,I(j,1,1)}, \dots, t^{\rho}_{j,I(i,u,r)}\big)$ instead of writing: $\mathsf{sk} = h\big(t^{\rho}_{i,I(j,1,1)}, \dots, t^{\rho}_{i,I(j,u,r)}, t^{\rho}_{j,I(i,1,1)}, \dots, t^{\rho}_{j,I(i,u,r)}\big)$.

ID_i knows:
$I(i,\mu,\rho), k^{\rho}_{I(i,\mu,\rho)}$ and T_i
$\mu \in [1,u], \rho \in [1,r]$

ID_j knows:
$I(j,\mu,\rho), k^{\rho}_{I(j,\mu,\rho)}$ and T_j
$\mu \in [1,u], \rho \in [1,r]$

$$\xrightarrow[\mu\in[1,u],\rho\in[1,r]]{I(i,\mu,\rho)}$$

$$\xleftarrow[\mu\in[1,u],\rho\in[1,r]]{I(j,\mu,\rho)}$$

$t_{j,I(i,\mu,\rho)} \leftarrow f(k^{\rho}_{I(i,\mu,\rho)}, \mathrm{ID}_j)$
$\mu \in [1,u], \rho \in [1,r]$
Read $t_{i,I(j,\mu,\rho)} = f(k^{\rho}_{I(j,\mu,\rho)}, \mathrm{ID}_i)$
from T_i s.t. $\mu \in [1,u], \rho \in [1,r]$
$\mathsf{sk} \leftarrow h\big(t^{\rho}_{i,I(j,1,1)}, \dots, t^{\rho}_{j,I(i,u,r)}\big)$

$t_{i,I(j,\mu,\rho)} \leftarrow f(k^{\rho}_{I(j,\mu,\rho)}, \mathrm{ID}_i)$
$\mu \in [1,u], \rho \in [1,r]$
Read $t_{j,I(i,\mu,\rho)} = f(k^{\rho}_{I(i,\mu,\rho)}, \mathrm{ID}_j)$
from T_j s.t. $\mu \in [1,u], \rho \in [1,r]$
$\mathsf{sk} \leftarrow h\big(t^{\rho}_{i,I(j,1,1)}, \dots, t^{\rho}_{j,I(i,u,r)}\big)$

Figure 4. Key exchange for the general case: $u > 1$ and $r > 1$

3 Adversarial Advantage

In this section, we consider an adversary who has corrupted n_c out of the n users and obtained their key material, *e.g.* by physically attacking the IoT devices containing those keys. The corruption can happen before the user gets the key material or afterwards; however, the main assumption of this section is that the indices of the stolen keys are random. We will consider targeted attacks in Section 4.3.

We compute probabilities and expectation values for the adversarial advantage as well as the optimal selection of security parameters in Section C for fixed memory.

3.1 Expected Number of Collected Keys

Let N_{key} be the number of distinct keys that the adversary gets on average after corrupting n_c users. Since two users may share keys, we get less than $u \times n_c$ keys per rack. The precise calculation is given below:

Lemma 1 (The expected number of keys obtained by the adversary). *Assuming that the adversary corrupts n_c users, the expected total number of distinct keys that the adversary holds is*

$$N_{\text{key}} = \ell \times \left(1 - \left(1 - \frac{u}{\ell}\right)^{n_c}\right).$$

The proof of Lemma 1 can be found in Appendix B.

3.2 Probabilities

We now consider the probability for the adversary to get a non-corrupted user's keys, *i.e.* that the targeted user's key indices are all among the key indices obtained from the corrupted users.

In the following, we denote by K a random variable taking values from 1 to ℓ. We let $K_a^i, i \in [1, n]$ and $a \in [1, u]$ be the random variables defining the key indices for user i (considering only one rack, *i.e.* $r = 1$). Note that these variables are not independent since we assume each user gets u distinct indices. Let $\mathcal{C}$ be the set of corrupted users and $\mathcal{H}$ the set of non-corrupted ones. We define $n_c := \text{card}(\mathcal{C})$ and $n_h := \text{card}(\mathcal{H})$, *i.e.* $n = n_c + n_h$.

Lemma 2 (The probability to get a targeted user's key). *With the above notations, for some given $i_0 \in \mathcal{H}$, the attacker's probability of getting a specified user's keys is denoted by $P_1 = P(\forall \mu \in [1, u] : K_\mu^{i_0} \in \{K_b^j\}_{j \in \mathcal{C}, b \in [1,u]})$, and the value is:*

$$P_1 = 1 - \sum_{i=1}^{u} (-1)^{i+1} \binom{u}{i} \prod_{j=1}^{i} a_j^{n_c}, \quad \textit{with} \quad a_j = \frac{\ell - j + 1 - u}{\ell - j + 1}.$$

For $r > 1$ we have

$$P_1 = \left(1 - \sum_{i=1}^{u} (-1)^{i+1} \binom{u}{i} \prod_{j=1}^{i} a_j^{n_c}\right)^r .$$

The proof of Lemma 2 is given in Appendix B. Based on the probability P_1, we can also find the probability of getting an arbitrary user's keys:

Lemma 3 (The probability to get an arbitrary user's key). *The probability of an attacker to get an arbitrary user's key, $P_2 = P(\exists i \in \mathcal{H} \ \ \forall a = 1, \dots, u : K_a^i \in \{K_b^j\}_{j \in \mathcal{C}, b \in [1,u]})$, is given by (also valid for $r > 1$):*

$$P_2 = 1 - (1 - P_1)^{n - n_c}.$$

The proof can be found in Appendix B.

Finally, we can consider the probability of getting the keys for two users, which will allow the attacker to break a session's keys.

Lemma 4 (The probability of getting two targeted users' keys). *The probability of the attacker to get two targeted users' keys and hence to break a shared key between them is $P_3 = (P_1)^2$.*

Lemma 5 (The probability to break an arbitrary session key). *The probability of an attacker to get two arbitrary users' keys and thus to break a session key is $P_4 = (P_2)^2$.*

Both Lemmas 4 and 5 follow directly from the independence of the allocated keys between users.

3.3 Optimising u

Given the probability P_1 defined in the last subsection, we can pose the question of whether there exists a non-trivial optimal value for u. To be specific, we fix a risk-level p and determine the maximal number of users that can be corrupted, n_c, while satisfying $P_1 \leq p$. The optimal value of u is the one allowing the largest amount of corrupted users, n_c. The problem is non-trivial since increasing u makes it harder for the adversary to get all keys from the targeted user, but on the other hand, the attacker gets more keys per corrupted user. Figure 5 shows that we indeed have non-trivial optimal values.

To make a precise analysis, we consider a large ℓ limit. A power expansion of P_1 gives

$$P_1 = \left(1 - \left(1 - \frac{u}{\ell}\right)^{n_c}\right)^u + \mathcal{O}\left(\frac{1}{\ell^2}\right).$$

We then observe that $n_c \sim \log(1 - P_1^{\frac{1}{u}}) / \log(1 - \frac{u}{\ell})$, *i.e.*

$$n_c/\ell \sim -u^{-1} \cdot \log(1 - P_1^{\frac{1}{u}}) ,$$

see Figure 5. We can use this expression to find the optimal value for u, forcing the adversary to corrupt as many possible users. By differentiation, we find the optimal u-value as

$$u = -\frac{\log P_1}{\log 2}.$$

To be able to have a risk level $P_1 = 2^{-m}$, the optimal u-value is $u = m$ and the adversary needs to corrupt approximately

$$n_c \sim -\ell \frac{\log^2 2}{\log P_1} = \ell \frac{\log 2}{m}$$

users. If we naively used $u = 1$, the attacker needs to corrupt

$$n_c \sim -\ell \log(1 - P_1) \sim \ell P_1 = \ell 2^{-m}$$

users to breach the risk level, where in the last approximation we assumed P_1 small. That is choosing the optimal u gives a significant advantage, actually logarithmic in the desired risk level. However, the flip side of increasing u is that the adversary has to corrupt fewer users to break the system entirely.

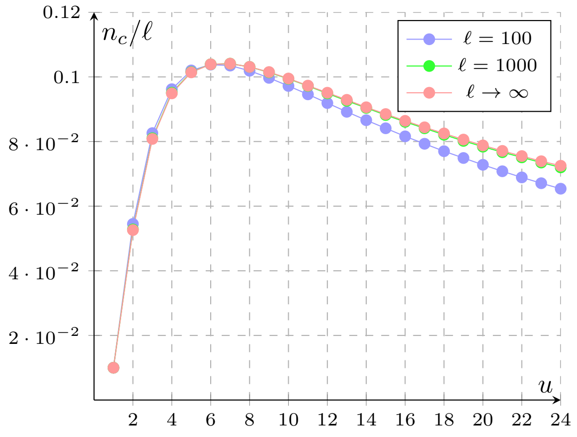

Figure 5. The relationship between u and n_c/ℓ for different values of ℓ ($\ell = 100, 1000, \infty$) and $P_1 = 1\%$; $\ell = 100, 1000$ have been found directly via the formula for P_1 in Lemma 2, whereas the curve for infinite ℓ plotted using the approximation above.

3.4 Expected Number of Corrupted Users to Full Breach – the Coupon Collector Problem

We now consider the expected number of users that the adversary needs to corrupt to reveal all keys, *i.e.* fully break the system. As discussed above, for $u = r = 1$ we have the classical coupon collector problem, where $n_c = \ell H(\ell)$ with H being the harmonic series.

For $u > 1, r = 1$ we clearly have $n_c \leq \ell H(\ell)/u$. The problem was analysed in the context of data package scheduling in [9] and the solution is $n_c = \sum_{i=0}^{\ell-1} \left(1 - \binom{i}{u}/\binom{l}{u}\right)^{-1}$. For large ℓ the speed-up is actually close to u: $\lim_{\ell\to\infty} \frac{n_c}{\ell \log \ell} = \frac{1}{u}$.

In the case $u = 1, r > 1$ we have only obtained an upper bound on n_c. Let n_i^ρ be the number of users needed to corrupt to get the i^{th} new key in rack ρ. Each n_i^r is geometrically distributed with probability parameter $p_i = (\ell - i + 1)/\ell$. Denoting the expectation value by E, for $r = 1$ we have that $n_c = E(\sum_{i=1}^{\ell} n_i) = \sum_{i=1}^{\ell} E(n_i) = \sum_{i=1}^{\ell} 1/p_i = \ell H(\ell)$ as mentioned above. For general r we have $n_c = E\left(\max_{\rho\in[1,r]}(\sum_{i=1}^{\ell} n_i^\rho)\right)$. However, even for the average of the maximum of geometric random variables, we do not have an explicit large r limit [6], only a closed sum formula. Nevertheless, using the bound for the maximum of geometric variables given in [6], we can get the following rough upper bound

$$n_c \leq \sum_{i=1}^{\ell} E\Big(\max_{\rho\in[1,r]}(n_i^\rho)\Big) \leq \sum_{i=1}^{\ell}\left(1 - \frac{H(r)}{\log(1-p_i)}\right) \leq \ell + \sum_{i=1}^{\ell}\frac{H(r)}{p_i} \leq \ell + H(r)\ell H(\ell)$$

Thus in limit $r \to \infty$ we see that $n_c/\ell \log \ell$ is bounded by $\log r$.

4 Security Analysis

The protocol can be seen as a special form of authenticated key exchange where the outcome is a fixed key. The authentication is implicit, *i.e.* Alice and Bob will hold the same key at the end of an undisturbed run of the protocol. In contrast, for an adversary who actively interrupts the communication and alters the transmitted indices, can make the keys might end up non-matching. However, the security guarantee we can give is an adversary who holds neither Alice's nor Bob's secret keys cannot distinguish the obtained secret key(s) from a random key.

Standard key-exchange protocols require complicated analyses of concurrent sessions. Nevertheless, in our case, we have a simple fixed protocol and we can split our analysis into three cases: ① a passive attacker only monitoring the communication, ② a man-in-the-middle attacker changing the information of the indices exchanged and, finally, ③ an active attacker trying to impersonate Alice or Bob.

Remark 5 (Explicit Authentication). For a protocol with explicit authentication to be achieved, an extra key-confirmation round can be added. One way to do this is by Alice sending $h_2(\text{sid}, \mathsf{sk})$ and Bob sending $h_3(\text{sid}, \mathsf{sk})$, where h_2 and h_3 are independent hash functions and the session identity sid contains Alice and Bob's ID and the indices exchanged. This is for one time use only; otherwise, we need to include nonces in the protocol to ensure freshness.

General Assumptions. Security, in general, relies on an honest setup ensured by a CA without information leakage (see, however, Section 4.4 on how to distribute the trust in CA). We also assume that the identities ID_i are publicly known and that they uniquely identify the users.

4.1 Passive Attacker

We start our analysis with the weakest attacker model, where the attacker can only observe the communication (*i.e.* see the indices $I(i)$ and $I(j)$ exchanged between participants ID_i and ID_j). Clearly, if the adversary holds the secret keys of both participants (*i.e.* $k^{\rho}_{I(i,\mu,\rho)}$ and $k^{\rho}_{I(j,\mu,\rho)}$ for all $\rho, \mu,$), then he will be able to reconstruct their secret key. We will now show that the obtained key is indeed indistinguishable from a random key for the adversary if he doesn't have all the keys.

We consider two cases. First, where the combiner function h is modelled in the ROM and f is EUF-CMA-secure.

Theorem 1. *Let the combining function h be modelled in the ROM and assume that f is an* EUF-CMA*-secure MAC. Then, a passive attacker can not distinguish the secret key sk obtained by ID_i and ID_j from a random key with a non-negligible probability unless he has obtained all of their keys $k^{\rho}_{I(i,\mu,\rho)}, k^{\rho}_{I(j,\mu,\rho)}$ for all $\mu \in [1, u], \rho \in [1, r]$.*

Proof. The secret key is $\mathsf{sk} = h\big(t^{\rho}_{i,I(j,1,1)}, \ldots, t^{\rho}_{i,I(j,u,r)}, t^{\rho}_{j,I(i,1,1)}, \ldots, t^{\rho}_{j,I(i,u,r)}\big)$. In the ROM this key can only be distinguished from random if the input value has been computed. This is only possible if all the MACs are either computed or already known by the adversary. Regarding the latter, the known tabulated MACs from the corrupted users are not useful since they contain the wrong ID. Thus the adversary has to compute the MACs which, by the EUF-CMA assumption, is only possible using the corresponding keys. If even a single key is unknown by the adversary, the probability of distinguishing sk from random is, thus, bounded by the advantage in the EUF-CMA game. Note that the adversary's known keys reduce the space of possible keys, since the keys in each rack are distinct, but for ℓ maximally polynomial in the key size, this is a negligible advantage.

Remark 6. The probability of breaking some session key for a static passive adversary or an adversary corrupting random users is given by P_4 and the probability to break an sk between two specific users is P_3.

Remark 7. Note that if the two users have the same index, the theorem still holds, but it is simply easier for the adversary to obtain all the keys.

Remark 8. The EUF-CMA assumption is too strong in the sense that we only need the adversary to be unable to compute the MAC of the identities. Even choosing f as a hash function of the ID and the key is safe in the ROM following the same proof structure.

Remark 9. It is also possible to relax the ROM and only consider h and f to be randomness extractors. This ensures that the adversary does not learn anything useful from the T_i tables of the corrupted users. Further, if just a single key is unknown, the obtained sk will still be indistinguishable from random.

4.2 Man-in-the-Middle and Authentication Attacks

We now consider an attacker who alters the sent messages, or even tries to pose as someone else to break authenticity. Note that in this case, we do not have any sk-security in the Canetti-Krawczyk model since the attacked users will not end up with the same key, but a key confirmation would help.

We also note that if the adversary gets all of Alice's keys, he can pretend to be any ID_j to Alice. The adversary simply sends an index, $I(j')$, from one of the corrupted users. Note that Alice is not supposed to keep a record of indices, so Alice will probably not detect that the wrong index is being sent. The adversary can now calculate sk using that all keys are known, and hence the MACs can be constructed.

Nevertheless, if the adversary is missing one of Alice's keys, he cannot distinguish the key computed by Alice from random.

Theorem 2. *Let the combining function h be modelled in the ROM and assume that f is* EUF-CMA*-secure MAC. Consider a user ID_i wanting to establish a key with ID_j. Even if the adversary alters the sent indices, he cannot distinguish the secret key sk obtained by ID_i from random with a non-negligible probability unless he has obtained all of the keys $k^{\rho}_{I(i,\mu,\rho)}$ for all $\mu \in [1, u], \rho \in [1, r]$.*

The proof follows as before, and all remarks about relaxing the assumption given in Section 4.1 also hold here.

Remark 10. An active adversary can thus successfully attack a specified user with probability P_1 and some arbitrary user with probability P_2.

4.3 Adaptive Corruption

In the protocol, the key indices are sent in clear. However, this is problematic in the case of adaptive attackers. If the adversary wants to target a specific user, he can then observe any key establishment to learn the index of that particular user. The adversary can then look for other users with the same index who might be easier to corrupt.

One possible countermeasure would be to use hybrid security techniques to make the indices private. Nonetheless, a more interesting approach would be to use the fact that both users entering into a key establishment already know that the resulting key will be one of ℓ different possible keys (here we take $r = u = 1$). As an example, ID_i wanting to talk to ID_j knows that the key is going to be $\mathsf{sk} = h(t_{i,I(j)}, t_{j,I(i)})$ and she can then simply compute all possibilities for $I(j) = 1, \dots, \ell$. The two users could hash their corresponding possibilities – the correct key will yield the same hash on both sides. They could now exchange these hashes in random order, and thus determine the shared key without revealing the indices. This could be done even with logarithmic efficiency.

4.4 The Central Authority

As our proposed protocol relies on a trusted third party (TTP), for analyzing security, we assume that the CA is not malicious. However, in real-life applications, this is not always the case. For example, due to the distributed nature of IoT devices, various dedicated authenticated key exchange protocols appeared in the literature. We are particularly interested in the results of [1] in terms of cryptographic layer separation and, more precisely, role distribution. Building on the model proposed in [1, Section 2.1] involving different roles for achieving different goals, we believe that distributing the power that a single CA normally has in a classical architecture can be useful especially in the context of our coupon-collector security-based protocol. As we introduced the idea of having r racks of keys, we may naturally distribute a rack per CA to minimize the security impact of a malicious third party. Nonetheless, other more exotic secret sharing schemes may be used to distribute the power between several CAs.

On another note, the idea presented in [7] bases its security on a TTP which *"also serves as an arbitrator when disputes arise due to a user denying certain actions"*. Besides relying on various CAs as previously mentioned, we stress that there are various methods of circumventing issues like trusting TTPs.

4.5 Post-Quantum Security

The primitives used in our proposed protocol (such as MACs and hash functions) seem to be good quantum-safe candidates. The main (optimal) quantum algorithm to break these is Grover's algorithm, which only gives a quadratic speed-up.

5 Conclusion and Further Development

We presented a new authenticated key exchange protocol entirely based on symmetric primitives and analyzed its security. We also discussed parameter choices and efficiency; we found especially interesting ways of improving security by handing out more keys per user while keeping memory usage constant.

Future Work. A natural research direction would be to formally analyze both the similarities of our proposed construction with standard public-key cryptography schemes and the post-quantum nature of our key distribution protocol. For a more precise security assessment, it is important to achieve better bounds for the expected number of corrupted users required to get a full breach in the case of general r, u – a problem which is an interesting coupon collector problem in its own right. It would also be interesting to understand in detail the u and r

duality phenomenon seen in Appendix C when dealing with constrained memory and a large ℓ.

Another possible venue of future research is to consider hybrids of the current protocol, *e.g.* by achieving forward secrecy relying on a computational assumption.

6 Acknowledgements

PBR and PYAR acknowledge support from the Luxembourg National Research Fund (FNR) under the FNR CORE project Q-CoDe, and the European Union's Horizon 2020 research and innovation programme under grant agreement No. 779391 (FutureTPM), in particular, Fatima-Ezzahra El Orche was supported by the Luxembourg National Research Fund through grant PRIDE15/10621687/SPsquared.

The authors would like to thank Natacha Laniado for her useful comments.

References

1. Avoine, G., Canard, S., Ferreira, L.: IoT-Friendly AKE: Forward Secrecy and Session Resumption Meet Symmetric-Key Cryptography. In: ESORICS'19. Lecture Notes in Computer Science, vol. 11736, pp. 463–483. Springer (2019)
2. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. http://cseweb.ucsd.edu/~mihir/papers/eakd.pdf (1993), full version
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: CCS'93. pp. 62–73. ACM (1993)
4. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Advances in Cryptology - CRYPTO'93. Lecture Notes in Computer Science, vol. 773, pp. 232–249. Springer (1994)
5. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (September 1976)
6. Eisenberg, B.: On the Expectation of the Maximum of IID Geometric Random Variables. Statistics & Probability Letters 78(2), 135–143 (2008)
7. Joye, M., Yen, S.M.: ID-based Secret-key Cryptography. SIGOPS Oper. Syst. Rev. 32(4), 33–39 (Oct 1998)
8. Needham, R.M., Schroeder, M.D.: Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM 21(12), 993–999 (1978)
9. Sharif, M., Hassibi, B.: Delay Considerations for Opportunistic Scheduling in Broadcast Fading Channels. IEEE Transactions on Wireless Communications 6(9), 3353–3363 (2007)

A Preliminaries

Notations. Throughout the paper, κ denotes a security parameter. We use the notation $x \in_R X$ when selecting a random element x from a sample space X. We denote by PPT a Probabilistic Polynomial-Time algorithm.

Random Oracles. Let RO represent the notion of a *random oracle*. Also, we denote by ROM the *random oracle model*. The widely adopted ROM was introduced in [3]. The model is characterized by either considering perfectly random functions or ROs (*i.e.* $\textcircled{1}$ each new query is returned a random answer and $\textcircled{2}$ if a given query is repeated it receives the same answer). Practical instantiations are usually done by means of hash functions.

Security of Message Authentication Codes. The experiment *existential unforgeability under chosen message attack* will further be denoted by EUF-CMA when referring to the security of message authentication codes (MACs). A MAC consists of three PPT algorithms $\mathtt{Setup}(1^\kappa), \mathtt{MAC}_k(m)$ and $\mathtt{Verify}_k(m, \text{tag})$.

We define the experiment $\mathtt{Exp}(n)^{\text{EUF-CMA}}_{\mathcal{A},\text{MAC}}$ by:

1. $k \leftarrow \mathtt{Setup}(1^\kappa)$;
2. $(m, \text{tag}) \leftarrow \mathcal{A}(1^\kappa)$. Let $\{m_i\}_1^q$ denote $\mathcal{A}$'s queries to $\mathtt{MAC}_k$;
3. If $\mathtt{Verify}(m, \text{tag}) = 1$ and $m \notin \{m_i\}_1^q$ return $\mathtt{Valid}$;
4. Otherwise return $\mathtt{Invalid}$.

Definition 1 (EUF-CMA). *A MAC consisting of the three algorithms* $\mathtt{Setup}$, $\mathtt{MAC}$ *and* $\mathtt{Verify}$ *is* EUF-CMA *(or simply secure) if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\mathtt{negl}$ *such that:*

$$\mathtt{Pr}[\mathtt{Exp}(n)^{\text{EUF-CMA}}_{\mathcal{A},\text{MAC}} = 1] \leq \mathtt{negl}(n).$$

Definition 2. *A* MAC *is considered* (t, ε)*-secure (*EUF-CMA*) if for all* t*-time adversaries* $\mathcal{A}$

$$\mathtt{Pr}[\mathtt{Exp}(n)^{\text{EUF-CMA}}_{\mathcal{A},\text{MAC}} = 1] \leq \varepsilon.$$

B Proofs from Section 3

B.1 Proof of Lemma 1

Proof. From the first user the adversary gets u keys. The second gives on average $u \times \left(1 - \frac{u}{\ell}\right)$ new keys since u keys are already taken. In general let N_i be the number of new keys gotten from the i^{th} user. We then have the average number of keys with n_c corrupted users

$$N_{\text{key}}(n_c) = E(\sum_i N_i) = \sum_i E(N_i) \ .$$

We note that we have a recursion

$$E(N_i) = u \times \left(1 - \frac{\sum_{j=1}^{i-1} E(N_j)}{\ell}\right) \ .$$

To see this let $p(k)$ be the probability of having k different keys just before the i^{th} corrupted users, that is $\sum_{j=1}^{i-1} E(N_j) = \sum_k k \cdot p(k)$. Given k keys the probability of getting m new keys is $\binom{u}{m} \left(\frac{l-k}{l}\right)^m \left(\frac{k}{l}\right)^{u-m}$. Thus

$$E(N_i) = \sum_{m=0}^{u} \sum_k m \binom{u}{m} \left(\frac{l-k}{l}\right)^m \left(\frac{k}{l}\right)^{u-m} p(k) \ .$$

Using standard differentiation methods, rewriting and solving we find that $\sum_{m=0}^{u} m\binom{u}{m} \left(\frac{l-k}{l}\right)^m \left(\frac{k}{l}\right)^{u-m} = u(1-\frac{k}{l})$, from which the relation follows.

We can rewrite the recursion as:

$$N_{\text{key}}(n_c) = N_{\text{key}}(n_c - 1) + u \times \left(1 - \frac{N_{\text{key}}(n_c-1)}{\ell}\right) \ ,$$

with the solution

$$N_{\text{key}}(n_c) = \ell \times \left(1 - \left(1 - \frac{u}{\ell}\right)^{n_c}\right) .$$

□

B.2 Proof of Lemma 2

Proof. For a given $i_0 \in \mathcal{H}$, we have:

$$\begin{aligned} P_1 &= P(\forall \mu \in [1,u] : K_\mu^{i_0} \in \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]}) \\ &= 1 - P(\exists \mu \in [1,u] : K_\mu^{i_0} \notin \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]}) \\ &= 1 - P(K_1^{i_0} \notin \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]} \text{ or } \ldots \text{ or } K_u^{i_0} \notin \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]}) \\ &= 1 - P_1' \end{aligned}$$

Let $A_i = \{K_i^{i_0} \notin \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]}\}$ for all $i \in [1,u]$.

Since $P_1' = P(A_1 \cup A_2 \cup \ldots \cup A_u) = \sum_{i=1}^{u} (-1)^{i+1} \binom{u}{i} P(A_1 \cap \ldots \cap A_i)$, it only remains to compute $P(A_1 \cap \ldots \cap A_i)$ for all $i \in [1,u]$ to complete the calculation of P_1:

$$\begin{aligned} P(A_1 \cap \ldots \cap A_i) &= P(K_1^{i_0} \notin \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]} \text{ and } \ldots \text{ and } K_i^{i_0} \notin \{K_\mu^j\}_{j \in \mathcal{C}, \mu \in [1,u]}) \\ &= \prod_{j=1}^{n_c} P(K_1^{i_0} \notin \{K_\mu^j\}_{\mu \in [1,u]} \text{ and } \ldots \text{ and } K_i^{i_0} \notin \{K_\mu^j\}_{\mu \in [1,u]}) \\ &= P(K_1^{i_0} \notin \{K_\mu^1\}_{\mu \in [1,u]} \text{ and } \ldots \text{ and } K_i^{i_0} \notin \{K_\mu^1\}_{\mu \in [1,u]})^{n_c} \\ &= \left(\frac{\ell - u}{\ell} \cdot \frac{\ell - u + 1}{\ell - 1} \cdots \frac{\ell - i + 1 - u}{\ell - i + 1}\right)^{n_c} \\ &= \prod_{j=1}^{i} \left(\frac{\ell - j + 1 - u}{\ell - j + 1}\right)^{n_c} \end{aligned}$$

The value for $r > 1$ follows from independence between the racks. □

B.3 Proof of Lemma 3

Proof. We have:

$$\begin{aligned}
P_2 &= P(\exists i \in H, \forall \mu \in [1,u] : K^i_\mu \in \{K^j_b\}_{j\in\mathcal{C}, b\in[1,u]}) \\
&= 1 - P(\forall i \in H, \exists \mu \in [1,u] : K^i_\mu \notin \{K^j_b\}_{j\in\mathcal{C}, b\in[1,u]}) \\
&= 1 - P(\bigcap_{i\in H} \{\{K^i_1, \dots, K^i_\mu\} \notin \{K^j_b\}_{j\in\mathcal{C}, b\in[1,u]}\}) \\
&= 1 - P(\{K^i_1, \dots, K^i_\mu\} \notin \{K^j_b\}_{j\in\mathcal{C}, b\in[1,u]})^{n-n_c} \\
&= 1 - (1 - P_1)^{n-n_c}
\end{aligned}$$

□

C Parameter Choice and Efficiency Analysis

In this section, we analyze the efficiency of our protocols based on the consideration of two types of attacks: small scale attacks and full breach attacks that we define in the next sections.
The user's global memory usage is determined by r and ℓ (as $r \times \ell$). Hence it is natural to fix $r \times \ell$ to some reasonable constant (*e.g.*, 1Mb) and assume that keys are 128 bits long (as in NIST's PQ-cryptography standardization). This implies that $r \times \ell = 2^{13}$. Thus the question boils down to finding the optimal u, r (and by implication the corresponding $\ell = 2^{13}/r$) maximizing n_c (the expected number of corrupted users) for a given n.

C.1 Low-Threat Scenario

Definition 3. *A* low-threat scenario *happens when the adversary succeeds to break the $u \times r$ keys of a (targeted or random) user with probability greater than or equal to ϵ, which we call later* the risk-level.

This section provides numerical values for lowering the adversary's success probability below ϵ. In the following, we consider the two different values of $\epsilon = 1‰,\ 0.01‰$ and we evaluate the attack probabilities found in Section 3.

Attack 1: Breaking the Keys of a Targeted User. This attack happens with probability P_1 which does not depend on n. Therefore, we are interested in finding the optimal u and r allowing maximizing n_c under the constraint:

$$P_1(\frac{2^{13}}{r}, u, n_c, r) \leq 1‰, 0.01‰$$

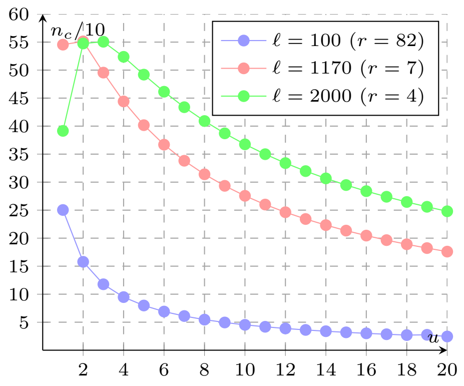

Figure 6. u and n_c for (ℓ, r) values s.t. $\ell r = 2^{13}$ and $P_1 = 1‰$.

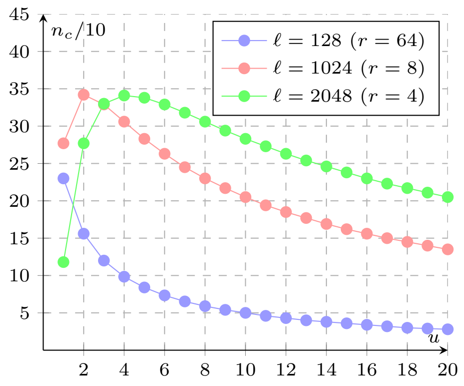

Figure 7. u and n_c for (ℓ, r) values s.t. $\ell r = 2^{13}$ and $P_1 = 0.01‰$.

Repeating the analysis from Section 3.3 for general u, r with a fixed memory size $\ell r = 2^{13}$, we find for large ℓ that the optimal parameter choice is reached for $ur = -\frac{\log P_1}{\log 2}$. We see clearly in Figure 6 and Figure 7 the presence of an optimum in some curves ($r = 4, 7$ in Figure 6 and $r = 4, 8$ in Figure 7). This optimum corresponds to a non-trivial optimal ur and it is increasing when P_1 is decreasing ($ur \sim 10$ for $P_1 = 1‰$ and $ur \sim 16$ for $P_1 = 0.01‰$). No optimum is noticed when $r > -\frac{\log P_1}{\log 2}$. Moreover, n_c reaches the highest values when the probability risk-level is large ($n_c^{\max} \sim 569$ for $P_1 = 1‰$ and $n_c^{\max} \sim 341$ for $P_1 = 0.01‰$).

To see whether we can differentiate the the parameters satisfying $ur = -\log P_1 / \log 2$, we further compute the expected number of corrupted users n_c needed for a full breach for the corresponding parameters (r, u, ℓ). We take $(r, u, \ell) = (r, -\frac{1}{r}\frac{\log P_1}{\log 2}, \frac{2^{13}}{r})$. Table 1 and Table 2 show the numerical values.

(r, u, ℓ)	$(1, 10, 2^{13})$	$(2, 5, 2^{12})$	$(5, 2, 1638)$	$(10, 1, 819)$
$n_c^{\max}$	7343	7885	7859	7853

Table 1. $n_c^{\max}$ to fully breach the system with $P_1 = 1‰$ and $\ell \times r = 2^{13}$

(r, u, ℓ)	$(1, 16, 2^{13})$	$(2, 8, 2^{12})$	$(4, 4, 2^{11})$	$(8, 2, 2^{10})$	$(16, 1, 2^9)$
$n_c^{\max}$	4929	4919	5065	4732	4928

Table 2. $n_c^{\max}$ to fully breach the system with $P_1 = 0.01‰$ and $\ell \times r = 2^{13}$

We notice from Table 1 and Table 2 that for all the possible optimal (r, u, ℓ) combinations, $n_c^{\max}$ always takes approximately the same value (well within the standard deviation of the Monte Carlo simulations used to obtain the tables) and only depending on the chosen P_1 level. Hence, the optimal combination (r, u, ℓ) is not *unique*. We conjecture that there is a duality between u and r with constrained memory. Note that we could try to explain this *e.g.* for $(r, u, \ell) = (1, 2, 2\ell) \mapsto (2, 1, \ell)$ by splitting a rack of size $2 \times \ell$ into two of size ℓ. However, two random keys from the original rack only have probability around

1/2 of being split into separate racks. Thus, further analysis is needed, which we postpone for future research.

Attack 2: Breaking the Keys of a Random User. This attack happens with probability P_2 which depends on n. Therefore, we are interested in finding the optimal u and r allowing maximizing n_c for a given n under the constraint:

$$P_2(\frac{2^{13}}{r}, u, n, n_c, r) \leq 0.1‰, 0.01‰$$

Tables 3, 4 investigate this for $n' = \log_{10}(n) = 1, \ldots, 6$. Values were obtained using a Python code.

	$u=1$	$u=2$	$u=3$	$u=4$	$u=5$
$n'=2$	$(9,7)$	$(2,57)$	$(1,60)$	$(1,67)$	$(1,69)$
$n'=3$	$(1,1)$	$(3,15)$	$(2,46)$	$(2,83)$	$(2,114)$
$n'=4$	$(1,1)$	$(3,5)$	$(2,21)$	$(2,45)$	$(2,69)$
$n'=5$	$(1,1)$	$(2,2)$	$(2,10)$	$(2,26)$	$(1,43)$
$n'=6$	$(1,1)$	$(1,1)$	$(2,5)$	$(2,15)$	$(1,27)$

Table 3. $(r^{\text{opt}}, n_c^{\max})$ for $P_2 = 1‰$

	$u=1$	$u=2$	$u=3$	$u=4$	$u=5$
$n'=2$	$(9,4)$	$(6,35)$	$(3,66)$	$(2,65)$	$(1,72)$
$n'=3$	$(1,1)$	$(4,9)$	$(4,28)$	$(3,50)$	$(2,70)$
$n'=4$	$(1,1)$	$(3,3)$	$(3,13)$	$(3,28)$	$(2,43)$
$n'=5$	$(1,1)$	$(4,2)$	$(4,7)$	$(3,16)$	$(2,27)$
$n'=6$	$(1,1)$	$(1,1)$	$(2,3)$	$(2,9)$	$(2,17)$

Table 4. $(r^{\text{opt}}, n_c^{\max})$ for $P_2 = 0.01‰$

From Table 3 and Table 4 we see that the highest value of n_c is reached when $(u, r, n) = (5, 2, 1000)$ ($n_c^{\max} = 114$) for $P_2 = 1‰$ and when $(u, r, n) = (5, 1, 100)$ ($n_c^{\max} = 72$) for $P_2 = 0.01‰$.

Remark 11. We notice that the value of $n_c^{\max}$ for Attack 1 is about 5 times bigger than the one for Attack 2 ($n_c^{\max}(P_1 = 1‰) = 569 > n_c^{\max}(P_2 = 1‰) = 114$ and $n_c^{\max}(P_1 = 0.01‰) = 341 > n_c^{\max}(P_2 = 0.01‰) = 72$).

C.2 Full Breach

Definition 4. *A* full system breach *happens when the adversary succeeds to recover all the $\ell \times r$ secret keys given by the CA.*

In the following, we are interested in finding the maximal value of n_c needed to fully breach the system and the corresponding r and u for a fixed memory size $M = \ell \times r = 2^{13}$. Table 5 shows the numerical values obtained after running Monte Carlo simulation in Python and taking $N = 1000$.

u	1	2	3	4	5	6	7
$n_c^{\max}$	40159	39270	25430	18323	15544	13283	10606
u	8	9	10	11	12	13	14
$n_c^{\max}$	10219	8413	7361	6884	6645	6326	5636

Table 5. Values of $n_c^{\max}$ to fully breach the system ($\ell \times r = 2^{13}$ and $N = 1000$). In all cases $r^{\text{opt}} = 1$.

$n_c^{\max}$ is strictly decreasing when u is increasing and reaches the highest value when $u = r = 1$.

C.3 Expected Number of Corrupted Users to Fully Breach the System

This section gives numerical values of n_c to fully breach the system. The following values are obtained using Monte Carlo simulation in Python and taking $N = 10000$.

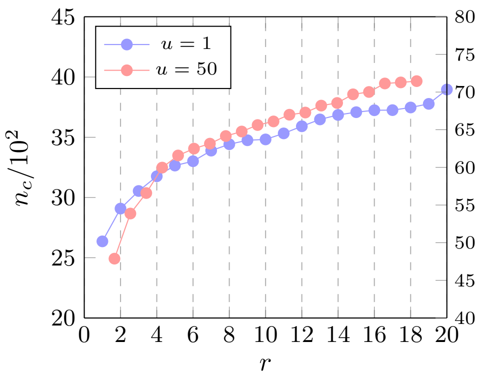

Figure 8. r and n_c for $\ell = 400$ and $u = 1, 50$ to recover all secret keys.

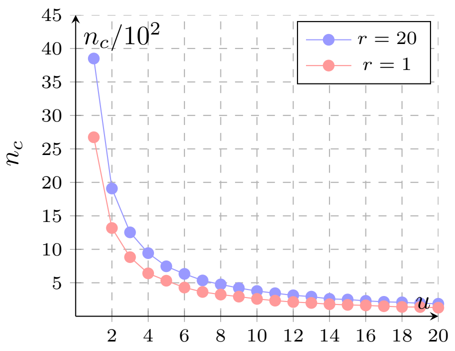

Figure 9. u and n_c for $\ell = 400$ and $r = 1, 20$ to recover all secret keys.

From Figure 8 and Figure 9 we see clearly that n_c is increasing when r is increasing and decreasing when u is increasing. The values obtained through this simulation are very close to the theoretical results of Section 3.4. We consider the main cases ($u = r = 1$, $u > 1, r = 1$ and $u = 1, r > 1$) and refer the reader to Table 6 for precise values.

Cases	$u = r = 1$	$u > 1, r = 1$	$u = 1, r > 1$
$(n_c^{\text{simu}}, n_c^{\text{theo}})$	$(2627, 2630)$	$(47, 47)$	$(3850 < 9854)$

Table 6. n_c values from Monte Carlo simulation and theoretical formulas