NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

CS4236 - Cryptography Theory and Practice

2021

The Projects for CS4236
(Cryptography Theory and Practice)
Singapore, November 2021.

Table of Contents

Security Formulation of Web Systems

Han Shutong[A0066762X], Ching Kian Weng, Kelvin[A0237553B], Tan Kian Chong[A0188653N], and Omprakash Padmanaban[A0243813J]

National University of Singapore

Abstract. This paper proposes MAC, Encryption schemes and pseudo-random random generater constructions (or mix of some scheme) to make web applications perfectly secure or computational secure against injection or information leaking. Their security properties are investigated and proven mathematically. Both network layer attacks and application layer attacks are investigated here. Finally, we will discuss about the modern security practices in the HTTPS context.

1 Introduction

In this paper, we will investigate formulations that examine specific notions of what security means for web systems. Therefore, our assumption is that the security of the web has yet to achieve perfect secrecy but it is computationally secure. In an ideal world, different web protocols and web applications are working with each other to provide computationally secure communication, but an adversary may still be able to bypass security through other means simply because of issues that cryptography alone is not sufficient to address. These issues such as vulnerable codes in web applications or poor input validation on websites can be exploited and lead to compromised servers.

However, it is not the intention of this paper to consider issues that are not within the sphere of control using cryptographic methodologies to provide secure communication for the web. Hence, we are specifically looking into prevention of information leak and injection. With that in mind, let us first look at formalism of the different security formulations before dwelling into modern web techniques to secure information transfer between client and server systems.

2 Security Formulation

2.1 HTTPS-injection-secure

Man-in-the-middle (MITM) attack could potentially introduce injection of HTTPS in-transit data causing the message integrity to be corrupted as the attacker gains the write access to the network traffic. To avoid the HTTPS injection, a message authentication code (MAC) scheme is adopted to authenticate the sender.

HTTPS injection secure scheme MAC (Gen, Mac, Verify) A message authentication code consists of three probabilistic polynomial-time algorithms:

1. The Gen takes as input the security parameter 1^n and outputs a key k with $|k| >= n$
2. The tag-generation algorithm Mac takes as input a key k and a message $m \in \{0,1\}^*$, and outputs a tag t. Since the algorithm maybe randomized, we write this as $t \longleftarrow Mac_k(m)$.
3. The deterministic verification algorithm Vrfy takes as input a key k, a message m and a tag t. It outputs a bit b with b = 1 meaning valid and b= 0 meaning invalid. We write this as $b := Vrfy_k(m,t)$

Perfectly Secure MAC Scheme We now define the perfectly secure MAC scheme. Intuitively, if no efficient adversary should be able to generate a valid tag on any message (including the old and new messages), then the construction is perfectly secure. To model the adversary's interaction with the communicating parties, we can have the following game, Mac-forge-no-oracle game:

1. A key k is generated by running Gen, and it is not given to adversary.
2. The adversary $\mathcal{A}$ produces m and output a tag t and send (m, t) to for verification.
3. The output of the experiment is defined to be 1 iff Vrfyk(m, t) = 1

A message authentication code $\Pi = (Gen, Mac, Vrfy)$ is perfectly secure if for all even unbounded adversaries $\mathcal{A}$, $Pr[Mac-forge-no-oracle_{A,\Pi} = 1] = 0$

However, this definition is impossible to achieve. Namely, we cannot hope to have a message authentication code for which the probability that an adversary outputs a valid tag on a previously unauthenticated message is 0. The reason is that an adversary can simply guess a valid tag t on any message and the guess will be correct with probability (at least) $1/2^{|t|}$, where $|t|$ denotes the tag length of the scheme. Hence, we relax the definition of the 'perfectly secure' MAC as follows:

A message authentication code $\Pi = (Gen, Mac, Vrfy)$ is slightly-less-than-perfectly secure if for all even unbounded adversaries $\mathcal{A}$, $Pr[Mac-forge-no-oracle_{A,\Pi} = 1] \leq negl(n)$

Construction of slightly-less-than-perfectly secure MAC Let $h : K \times M \longrightarrow T$ be a strongly universal function. Define a MAC for messages in M as follows.

- Gen : Choose uniform $k \in K$ and output it as the key
- Mac : on input a key $k \in K$ and a message $m \in M$, output the tag $t := h_k(m)$
- $Vrfy$: on input a key $k \in K$ and a message $m \in M$, and a tag $t \in T$, output 1 iff $t == h_k(m)$. If $m \notin M$ then output 0.

Theorem 1. *Let $h : K \times M \longrightarrow T$ be a strongly universal function. Then Construction above is a slightly-less-than-perfectly secure MAC for $m \in M$*

Proof. $Pr[Mac-forge-no-oracle_{A,\Pi}=1] = \sum_{t' \in T} Pr[Mac-forge-no-oracle_{A,\Pi}=1 \wedge h_k(m')=t']$
$= \sum_{t' \in T;(m,t):=A(t')} Pr[h_k(m)=t \wedge h_k(m')=t']$
$= \sum_{t' \in T;(m,t):=A(t')} 1/|T|^2 = 1/|T| = negl(n)$

In the context of HTTPS, communicating parties have to agree on the K where each k in K is used only for one-time and requires the length to be at least 2n (or a very long key). Therefore, we will now explore other constructions that is relaxed in the security level but much more practical to achieve.

Computational Secure MAC Scheme In the HTTPS protocol, it is impractical to having variable length key depending on the length of message, we propose another construction of MAC scheme that simulates the adversary as MITM attacker using a fixed length key. We can assume that the attacker can have access to a MAC oracle since the client can be compromised through phishing attacks which is easily achievable through MITM attack.

To model the adversary communication with different parties, we propose a game Mac-forge experiment:

1. A key k is generated by running Gen(1^n).
2. The adversary $\mathcal{A}$ is given input 1^n and oracle access to $Mac_k(.)$ and eventually outputs (m, t). All queries that A asked its oracle has been stored in Q.
3. $\mathcal{A}$ succeeds (or experiment outputs 1) iff $m \notin Q$ and $Vrfy_k(m,t) = 1$

We define that a message authentication code $\Pi = (Gen, Mac, Vrfy)$ is computational secure under MITM attack if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a negligible function negl such that: $Pr[Mac-forge_{A,\Pi}(n)=1] \leq negl(n)$

Construction of Computational Secure MAC:

Let F be a pseudorandom function and message has length n as follows:

- Mac : on input key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, output the tag $t := F_k(m)$ if $|m| \neq |k|$ then output will be nothing.
- $Vrfy$: on input key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, and a tag $t \in \{0,1\}^n$, output 1 iff $t == F_k(m)$ if $|m| \neq |k|$ then output 0.

Theorem 2. *If F is a pseudorandom function, then above construction is a secured MAC for messages of length n*

Proof. Let $\mathcal{A}$ be a probabilistic polynominal-time adversary. Consider a similar construction $acute\Pi = (Gen, Mac, Vrfy)$ same as above except that instead of using PRF, the system uses the real random number. $Gen(1^n)$ chooses function $f \in Func_n$ and Mac computes a tag t just as Mac which except that f is used instead of F_k that implies.

$Pr[Mac-forge_{A,\acute{\Pi}}(n)=1] \leq 2^{-n}$

from here we can show that negligible function negl as

$Pr[Mac - forge_{A,\Pi}(n) = 1] - Pr[Mac - forge_{A,\acute{\Pi}}(n) = 1] \leq negl(n)$

We write as combined equation as

$Pr[Mac - forge_{A,\Pi}(n) = 1] \leq 2^{-n} + negl(n)$

Firstly, we construct polynominal-time distinguisher D. D is given input 1 and access to an oracle $O : \{0,1\}^n \longrightarrow \{0,1\}^n$ and so on and so forth The view of adversary $\mathcal{A}$ when run sub-routine by D is distributed identically to the view of $\mathcal{A}$ in our experiment $Mac - forge_{A,\Pi}(n)$, and D output as 1 when $Mac - forge_{A,\Pi}(n) = 1$. Let's derive the equation from here,

$Pr[D^{F_k(.)}(1^n) = 1] = Pr[Mac - forge_{A,\Pi}(n) = 1]$,

Where $k \in \{0,1\}^n$ is chosen uniformly in the above

$Pr[D^{f(.)}(1^n) = 1] = Pr[Mac - forge_{A,\acute{\Pi}}(n) = 1]$,

Since F is a pseudorandom function and D runs in polynomial time there already exists negl as,

$|Pr[D^{F_k(.)}(1^n) = 1] - Pr[D^{f(.)}(1^n) = 1]| \leq negl(n)$

This ensures secure MAC that an adversary cannot generate valid tags t on new messages that was never authenticated. Upon forge modification tag t can be compromised. The proof settings useful to consider stronger definition of security for MAC where attack will be ruled out.

We also can consider if the construction is also feasible for a modified game with stronger definition of security, Mac-strong-forge game:

1. A key k is generated by running Gen(1^n).
2. The adversary $\mathcal{A}$ is given input 1^n and oracle access to $Mac_k(.)$ and eventually outputs (m, t). All queries that $\mathcal{A}$ asked its oracle has been stored in Q.
3. $\mathcal{A}$ succeeds (or experiment outputs 1) iff $(m,t) \notin Q$ and $Vrfy_k(m,t) = 1$

Theorem 3. *Since the construction is a secure MAC and the Verification process is Canonical Verification (we are recompute the tag to check the equality and MAC is deterministic), this construction is also a strongly secure MAC scheme.*

Proof. If a system uses canonical verification then Mac is deterministic. The only difference in the experiments Mac-forge and Mac-strong-forge is that, on output (m, t), the first one checks in the end whether m was already queried, whereas the latter checks whether m was already queried and additionally the result of the query was t. Give that Mac is deterministic, each query of any message m produces a unique tag t. So "m was already queried" is equivalent to "m was already queried and the output tag was t". Therefore, the experiments are equivalent for MACs that use canonical verification which implies that a secure MAC that uses canonical verification is also a strong secure MAC.

2.2 HTTPS-eavesdrop-secure

HTTPS-eavesdrop attack is successful if attacker could decrypt the encrypted in-transit data. We assume that the attacker has gained control over the network traffic through relay. The attacker could listen to the network traffic and get

the bits in-transit. The encrypted message is refering to the content of HTTPS document in the ciphertext form. We can assume that the scheme of encryption is private key encryption scheme where key is not known to the attacker. We will discuss different possible schemes and the secure definition of it.

HTTPS-eavesdrop perfectly secure encryption scheme We know that the HTTPS communicates through symmetric encryption. The HTTPS encryption scheme (Gen,Enc,Dec) with message space M is perfectly secret if for every probability distribution over M, every message m ∈ M, and every ciphertext c ∈ C for which Pr[C = c] > 0: $Pr[M = m|C = c] = Pr[M = m]$

To achieve that, we can define the construction as follows:

- Gen: the key-generation algorithm chooses a key from K = $\{0,1\}^l$ according to the uniform distribution.
- Enc: given a key k *in* $\{0,1\}^l$ and a message m *in* $\{0,1\}^l$, the encryption algorithm outputs the ciphertext c $\longleftarrow k \oplus m$.
- Dec: given a key k *in* $\{0,1\}^l$ and a ciphertext c *in* $\{0,1\}^l$, the decryption algorithm outputs the message m $\longleftarrow c \oplus k$.

In practice, for each pair of client and server, they should agree on n blocks of keys of size l beforehand. The messages are also sent in blocks of size l. Each block of message is encrypted by using a unique block of key in sequence. If all keys are used, client and server should re-negotiate the block of keys.

Theorem 4. *The HTTPS-eavesdrop perfectly secure scheme as described above is perfectly secret.*

Proof. $Pr[M = m|C = c] = Pr[C = c|M = m] * Pr[M = m]/Pr[C = c]$
$= Pr[Enc_k(m) = c] * Pr[M = m]/2^l = 2^l * Pr[M = m]/2^l = Pr[M = m]$

To model this scheme in terms of experiments, we define an experiment, called HTTPS-eavesdrop perfectly secure experiment described in Fig 1.

HTTPS Computational Secure Encryption Scheme Perfect secrecy encryption scheme is not a practical HTTPS encryption scheme because perfect secrecy is only possible when key is at least same size as message. However in reality, HTTPS needs to send very long messages and key cannot be very huge. Thus, a more practical encryption scheme will be having fixed length key and use the same key for encryption for a given M.

By definition, a scheme is computational secure if for every probabilistic polynomial time adversary $\mathcal{A}$ carrying out an attack, the probability that $\mathcal{A}$ succeeds in the attack is negligible.

We assume that the key is randomly generated. The encryption algorithm is known but key is unknown to the attacker. The attacker also can control the network traffic (read and write access). As an attacker in the context of HTTPS protocol, he/she can perform chosen plaintext attack (CPA attack) since the

attacker can perform phishing attacks to the client unlimited times so that client can send as many desired queries to server as possible. At same time, we also need to consider that the encryption scheme should be secure under multiple encryptions However, it is hard for the attacker to perform chosen ciphertext attack (CCA attack) as the server or client's decrypted message is not visible by listening to network traffic. Therefore, we need to develop a construction that is secure against the multiple encrytion and CPA attack so that the HTTPS eavesdrop attack can be avoided.

The construction of HTTPS eavesdrop attack under multiple encryption secure scheme is described as below:

Let F be a pseudorandom function. Messages can be broken down into blocks. Each block, m, has size l.

- Gen: the key-generation algorithm chooses a key from K = $\{0,1\}^l$ according to the uniform distribution.
- Enc: given a key k *in* $\{0,1\}^l$ and a message block m *in* $\{0,1\}^l$, choose uniform r *in* $\{0,1\}^l$ the encryption algorithm outputs the ciphertext $c :=< r, F_k(r) \oplus m >$.
- Dec: given a key k *in* $\{0,1\}^l$ and a ciphertext $c :=< r, s >$., the decryption algorithm outputs the message $m := F_k(r) \oplus s$.

In practice, the messages of any size i will be broken into $\lceil i/l \rceil$ blocks. Each block will have a size of l. A key k of length l is generated first and will be used for encryption of all message blocks. Before encryption of each block, r is generated. Then encryption of a message block will output c. The client or server can concatenate ciphertext, c, from all message blocks together and send them over to the opposite party.

To model the https-eavesdrop attack (e.g. CPA attack) with multiple encryption, we can have the following experiment for HTTPS eavesdrop attack under multiple encryption secure scheme $\Pi = (Gen, Enc, Dec)$, adversary $\mathcal{A}$, and value n for the security parameter.

1. A key k is generated by running $Gen(1^n)$

2. A uniform bit $b \in 0, 1$ is chosen by the challenger.

3. Adversary $\mathcal{A}$ is given input 1^n and oracle access to $LR_{k,b}(\cdot,\cdot)$. This LR oracle will take in two messages m_0, m_1 of the same length, and output the encrypted message from any one of the input based on the choice of b.

4. The adversary $\mathcal{A}$ outputs a bit b'.

5. The output of the experiment is defined to be 1 if b' = b, and 0 otherwise. If the output of the experiment is 1, we say that $\mathcal{A}$ succeeds.

Theorem 5. *The HTTPS eavesdrop attack under multiple encryption secure scheme as described above is computational secure, equivalent to CPA secure. For all probabilistic polynomial-time adversary A there is a negligible function negl such that* $Pr[PrivK^{cpa}_{A,\Pi}(n) = 1] \leq 1/2 + negl(n)$

Proof. Given the L-R oracle, the adversary $\mathcal{A}$ could construct a PPT distinguisher D to help $\mathcal{A}$ with guessing b. Let's consider the best polynomial time D we can think of:

The distinguisher D can query the message (m_0, m_0) repeatedly in polynomial time (e.g. p(n)), where message is of size l and distinguisher records all unique outputs. After up to p(n) unique record obtained, we can use (m_0, m_1) to feed into oracle, and check if the record contains the output, $Enc_{k,b}(m_b)$. If the records contains $Enc_{k,b}(m_b)$, then output 0, else output 1. Consider the best scenario where each query to oracle produce a unique record. $Pr[b' = 0] = p(n)/2^l$; $Pr[b' = 1] = 1 - p(n)/2^l$

The chance of guessing b correctly is: $Pr[b' = 1 \wedge b = 1] + Pr[b' = 0 \wedge b = 0]$

If output is 0, then it is guaranteed that b = 0 is also true. Therefore,

$Pr[b' = 0 \wedge b = 0] = Pr[b = 0|b' = 0] * Pr[b' = 0] = 1 * Pr[b' = 0] = Pr[b' = 0]$

Since the occurrence of b'=1 includes the occurrence of b = 1, given that b = 1, b'=1 must be true. Therefore,

$Pr[b' = 1 \wedge b = 1] = Pr[b' = 1|b = 1] * Pr[b = 1] = 1 * Pr[b = 1] = Pr[b = 1]$

Combining what we have:

$Pr[b' = 1 \wedge b = 1] + Pr[b' = 0 \wedge b = 0] = Pr[b = 1] + Pr[b' = 0]$

$= 1/2 + p(n)/2^l$ where $p(n)/2^l$ is negligible.

If we consider other scenarios where query results could be duplicated, $Pr[b' = 1 \wedge b = 1] + Pr[b' = 0 \wedge b = 0] <= 1/2 + negl(n)$

HTTPS Computational Secure Encryption-Authentication Scheme In the previous sections, eavesdrop attack and injection attack are considered as separate attacks and independent scheme are proposed for each attack. In fact, authentication scheme (MAC scheme) implemented for defensing injection attack can enhance the security of the encryption scheme.

The combined scheme is called authenticated encryption scheme. In particular, we have encrypt-then-authenticated scheme construction:

- Gen: on input 1^n chooses independent keys kE, kM $\in \{0,1\}^n$ according to the uniform distribution.
- Enc: given keys (kE, kM) and a message block m in $\{0,1\}^n$, choose uniform r in $\{0,1\}^n$ the encryption algorithm outputs the ciphertext $c :=< r, F_{kE}(r) \oplus m >$ and $t := Mac_{kM}(c)$. Finally, output the ciphertext $< c, t >$.
- Dec: given keys (kE, kM) and a ciphertext $< c, t >$, first check whether $Vrfy_{kM}(c, t) == 1$. If yes, the decryption algorithm outputs the message $m := F_{kE}(r) \oplus s.$, where s is part of c.

To model the construction, we can have a game like the Fig. 2. The key is generated first. Adversary $\mathcal{A}$ has an encryption oracle and output ciphertext c. Let m' be the decrypted message of new c. Q is the set of all queries $\mathcal{A}$ asked. $\mathcal{A}$ wins iff m' is not null (verification passes) and m' is not part of Q.

Theorem 6. *HTTPS Computational Secure Encryption-Authentication Scheme is also secure against chosen-ciphertext attacks.* $Pr[Awins] <= 1/2 + negl(n)$

Proof. We know that the construction is combination of strong mac experiment and CPA secure experiment. Since we know that for the strong mac experiment the probability of winning is less than negl(n), we can conclude that the event

of $\mathcal{A}$ submits a valid ciphertext to its decryption oracle (in CCA experiment) should be $Pr[AValidQuery]/p(n) <= negl(n)$

Since we know that for the CPA secure experiment, the chance of winning is less than 1/2 + negl(n), we can conclude that $Pr[Awins \wedge notAValidQuery] <= 1/2 + negl(n)$

Therefore combining both equations, we can have $Pr[Awins] <= 1/2 + negl(n)$

2.3 Session Cloning Attack

Session cloning attacks involve hijacking the current session of another user by using the ID of that session to gain unauthorised access to information or services in a system. This is typically done through the use of a cookie which stores the session ID. The attacker will insert the session ID of another user's into their own cookie in order to hijack that session.

Here we will define what it means to have a system that is secure from such an attack, or in other words, session cloning attack secure.

Session Cloning Attack Secure Definition We take the length of a cookie to be n. For simplicity, let us assume that for each bit of a cookie, $b_i \in 0, 1$. Take V to be the set of all valid cookies, and x to be an attacker's guess on the cookies. We say that the system is computationally secure from session cloning attacks if:
$Pr[x \in V] \leq negl(n)$ i.e. the chance of an attacker obtaining a valid cookie is at most negligible, or computationally difficult.

In addition, we say that the system is perfectly secure from session cloning attacks if: $Pr[x \in V] = \frac{|V|}{2^n} = 0$ that is, the chance of an attacker obtaining a valid token is equal to 0. Apparently, to achieve that we need n to be infinity. Therefore perfectly secure cookie is impossible in practice given that cookie has some size limit.

Cookie without Signature Scheme If the cookie structure contains only the session id of size n which is generated using a randomized function, we can define that if the session id not predictable it is session cloning attack secure. In other words, if no one can tell if the session id was generated randomly, then it is not predictable which gives us a secure session id. Let's take a distinguisher, D, a web system, S, and an arbitrary user session, U. We assume that D has unlimited excess to an oracle which generates session id. We propose a session id indistinguishability(SII) experiments:

1. D is given b, a session id from U of length n, and c, a uniformly chosen string data of size n, where $b \neq c$.
2. D outputs $x \in b, c$. The output of the test is 1 if $x = b$, 0 otherwise.

Session IDs from S are considered computationally indistinguishable if
$Pr[SII_{D,S,U} == 1] \leq 1/2 + negl(n)$

Theorem 7. *If session IDs are computationally indistinguishable, the system is computationally secure.*

Proof. Suppose the system is not computationally secure. Then,
$Pr[x \in V] = p, p > negl(n)$
Choose x_0 such that $Pr[x = x_0] = max(Pr[x = x_i]), x_i \in V$. Similarly, choose y_0 such that $Pr[y = y_0] = min(Pr[y = y_i]), y_i \in V^c$,

$Pr[x = x_0] > \frac{Pr[x \in V]}{|V|} = \frac{p}{|V|} > \frac{\frac{|V|}{2^n}}{|V|} + negl(n) = \frac{1}{2^n} + negl(n)$

$Pr[y = y_0] < \frac{Pr[y \in V^c]}{|V^c|} < \frac{1}{2^n}$

$Pr[x = x_0 | x = x_0 \vee y_0] = \frac{\frac{p}{|V|}}{\frac{p}{|V|} + Pr[y = y_0]} > \frac{1}{2} + negl(n)$

Theorem 8. *Computational security does not imply computational indistinguishability.*

Proof. Computational Security $\wedge \neg$ Computational Indistinguishability:
Consider a probability distribution such that $Pr[x = b] = \frac{2}{2^n}, b \in V$ but $Pr[x = c] = \frac{1}{2^n}, \forall c \notin V, c \neq 0^n$, and $Pr[x = 0^n] = 0, 0^n \notin V, |V| = 1$.
Then,$Pr[x \in V] = \frac{2}{2^n} \leq negl(n)$ but $Pr[x = b | x = b \vee x = c] = \frac{\frac{2}{2^n}}{\frac{2}{2^n} + \frac{1}{2^n}} = \frac{2}{3} > negl(n)$
Similarly, perfect security does not imply perfect indistinguishability.

Cookie with Signature Scheme We propose a cookie structure that contains 2 main things - data containing the session ID and a signature of the data, signed with a MAC. Each user should have a unique key to produce signature. Although attacker, who is one of the user of the web application, can send unlimited number of forged cookies to the server, each user is using a different key to sign the cookie. Hence, there is no oracle access available if client is not compromised because oracle requires to know the key from the victim. We propose the following cookie forge experiment for formalising the definition:

- $\mathcal{A}$ outputs a signed cookie c of size 2n given that A knows how to do MAC.
- $\mathcal{A}$ send the cookies to server.
- Server will check if the cookie is valid. If the cookie is valid (valid signature and valid session id), then the experiment output 1; else output 0

We say that a system is computational secure under the brute force attack if, for any system, $Pr[Cookie - forge_{A,G}] \leq negl(n)$

Proof. $Pr[Cookie - forge_{A,G}] = Pr[validSessionId \& validSignature]$
$= 1/p(n) * 1/p(n) = 1/p(n)^2 \leq negl(n)$

However, if the victim has been compromised (such as under phishing attack), we can have unlimited access to MAC oracle. We also know that the session id varies for the same user when they access to the web application at different sessions. Then we can have the another version of cookie-forge game to simulate session cloning attack with such condition.

1. $\mathcal{A}$ has MAC oracle access. $\mathcal{A}$ outputs (m, t), where m and t are the message and its signature respectively. Let Q be all the messages and keys, the adversary had sent to the oracle in this experiment.
2. Adversary wins (output of experiment is 1) iff (m, t) is valid and $m \notin Q$.

Similarly, we say that the system is strong cookie-forge secure under similar conditions, but $(m, t) \notin Q$.

Theorem 9. *If the MAC is a secure MAC, then the system is cookie-forge secure. Similarly, if the MAC is strongly secure, the system is strongly cookie-forge secure.*

Proof. $Pr[Mac - forge_{A,G} = 1] = Pr[(m, t)|m \notin Q]$
$= Pr[Cookie - forge_{A,G}] \leq negl(n)$
where m, t is any valid message signature pair.

Theorem 10. *If the session ID is not predictable, the system is cookie-forge secure.*

Proof. $Pr[Cookie - forge_{A,G}] = Pr[(m, t) = (m_1, t_1)|(m_1, t_1) \in V, m_0 \notin Q]$
$\leq Pr[m = m_1|m_1 \in V] \leq negl(n)$

Theorem 11. *If the system is cookie-forge secure, then the system is computationally secure from session cloning attacks.*

Proof. $Pr[x \in V] = Pr[(m, t) = (m_1, t_1)|(m_1, t_1) \in V, m_0 \notin Q] =$
$Pr[Cookie - forge_{A,G}] \leq negl(n)$

3 Security Practices and Techniques

3.1 Techniques to prevent system information leak

Modern day web systems rely on Certificate Authorities, Transport Layer Security (TLS) and Public Key Cryptography to establish trust and security with different users before any exchange of communication is carried out between the client and server. These components work together to prevent systems from leaking information to the adversary. Certificate Authorities (CA) are trusted third parties that are part of the Public Key Infrastructure (PKI) that issues verified digital certificates such that a client could use it to check whether a domain server is genuine or malicious prior to any connection made [1]. This is verified by decrypting the digital signature of the requested server with the server's public key that is stored within the certificate issued by the trusted CA.

If the digital signature is not signed with the corresponding private key from the server, the client's web browser will not connect to the domain and alert the user about the credibility of the server. CA actively track certificates that are no longer valid for use and add them into the Certificate Revocation List (CRL) and users will be able to make ad hoc query via the Online Certificate Status Protocol (OCSP) to verify against expired or insecure certificates and detect spoofed certificates. This prevents Man in the Middle (MITM) attacks whereby an adversary could pretend to be a trusted domain that a user requested and subject the user to a malicious web address to carry out exploits.

The Hyper Text Transfer Protocol Secure (HTTPS) uses the Transport Layer Security (TLS) to encrypt communication between client and server across the internet [2]. TLS is designed to prevent information leak by establishing the necessary handshakes between client and server wherever the client requests to access the server's domain [3]. During the initial handshakes, the parties need to first agree on which algorithms to use from the cipher suite [4]. To facilitate identity verification and secure information exchange, client-server applications use Public Key Cryptography to establish secure communication which uses the public-private key pairs to encrypt and decrypt information between the sender and recipient [5]. RSA is one of the most widely used public key cryptosystem between the client and server in web systems [6] and the algorithm's entire security depends on the difficulty of factoring and finding the two prime numbers, P and Q that were used to compute the product of a very large number N.

In addition to the security provided by public key cryptosystem, session keys are often exchanged in the process between client and server for symmetric encryption to take place. Session keys have limited life span of usage and after a period of inactivity in communication between client and server, they become invalid, and a new session key needs to be generated to communicate again. Forward secrecy is achieved through this method because if the current session key is compromised, the adversary cannot use it to decrypt past messages. For the session key to remain secure against attacks, the generated session key needs to be pseudorandom to avoid prediction attack which can happen for keys that are generated in a deterministic way. Many modern-day web systems use the Advanced Encryption Standard (AES) algorithm which has a minimal key length of 128 bits to perform symmetric encryption based on substitution-permutation networks and AES has remained computationally secure against brute-force attacks and no practical attacks were found to be able to break AES when implemented correctly [7,8].

3.2 Techniques to prevent injection into systems

Apart from eavesdropping on leaked communication, alteration of transmitted information between client and server is another kind of attack that can be carried out by the adversary. To prevent such attacks, modern day web systems use functions like Secure Hash Algorithm (SHA) to create hash based message

authentication codes (HMAC) to verify the integrity of the data [9]. This can be done using a shared secret session key that is known only between the client-server to generate a hash output value which will be sent along with the data for the recipient to verify. If the recipient can output the exact same hash value using the shared secret session key and agreed hash function, then the integrity of the data has not been compromised.

This method of verification requires the use of collision resistant hash algorithms that has negligible probability of two different sets of data, x and x', that produce the exact same hash value, $H(x) = H(x')$. Without such a property, it is possible that the adversary may obtain a collision hash value to manipulate the integrity of the data. Such an occurrence has proven to be possible with the widely used Message Digest algorithm 5 (MD-5) which produces a 128 bit hash value from data inputs of any size. Research had concluded that it is possible to obtain collision hashes in MD-5 in under a minute using ordinary computer machines [10]. Therefore, choosing a secure hash algorithm such as SHA-256 bits or SHA-512 bits which output a longer hash value can prevent injection attacks from succeeding.

4 Conclusion

Security in web systems needs to continuously evolve with the discovery of flaws that were found in the application of cryptography. In the last two decades, we have seen the deprecation of earlier versions of Secure Socket Layer (SSL 2.0/3.0) and Transport Layer Security (TLS 1.0/1.1) which were the main standards that the internet rely on for secure encrypted communications. With digitalization and increase reliance on the internet for business, essential services and considering the rise of quantum computing that may reduce the existing strength of cryptographic techniques, security formulation for web systems will remain an important work in the future to help the internet be a trusted platform for all users.

References

1. Certificate Authority, https://en.wikipedia.org/wiki/Certificate_authority.
2. HTTPS, https://en.wikipedia.org/wiki/HTTPS.
3. Transport Layer Security, https://en.wikipedia.org/wiki/Transport_Layer_Security.
4. Cipher Suite, https://en.wikipedia.org/wiki/Cipher_suite.
5. Public Key Cryptography, https://en.wikipedia.org/wiki/Public-key_cryptography.
6. RSA, https://en.wikipedia.org/wiki/RSA_(cryptosystem).
7. SPNetwork, https://en.wikipedia.org/wiki/Substitution-permutation_network.
8. AES, https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.
9. HMAC, https://en.wikipedia.org/wiki/HMAC.
10. MD-5 collision within a minute, https://if.fm/3kyVS.
11. Session hijacking attack, https://owasp.org/www-community/attacks/Session_hijacking_attack

A Appendix Diagrams

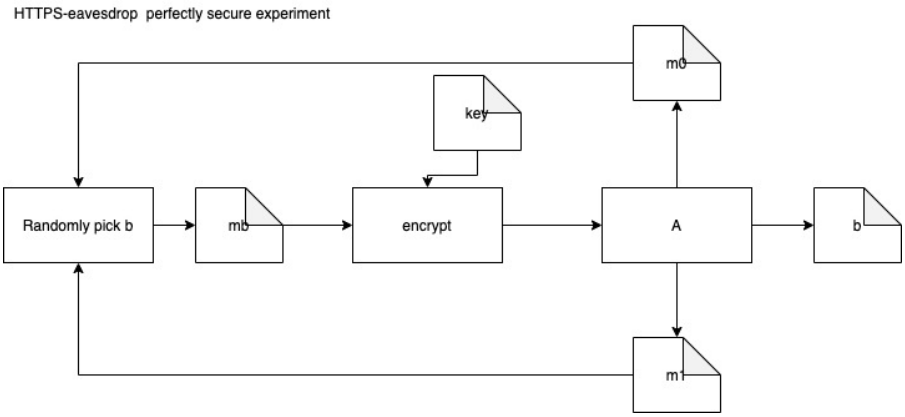


Fig. 1. HTTPS-eavesdrop perfectly secure experiment

1. The adversary $\mathcal{A}$ outputs a pair of messages $m_0, m_1 \in M$
2. A key k is generated using Gen and a uniform bit $b \in \{0, 1\}$ is chosen. Ciphertext c is computed by $Enc_k(m_b)$ and return c to $\mathcal{A}$.
3. $\mathcal{A}$ outputs a bit b'
4. If b' = b the experiment outputs 1, else output 0.

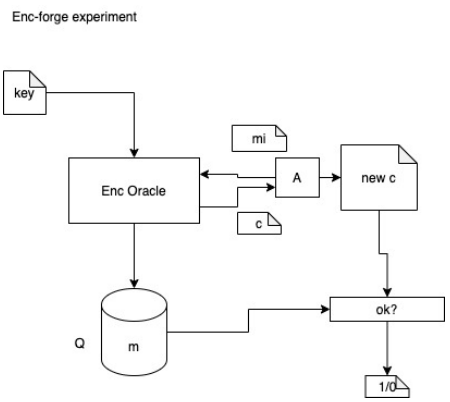


Fig. 2. The HTTPS authenticated encryption scheme experiment

Security of Contact Tracing protocols using Bluetooth Technology

Chong Sidney[1], Divakar Malhotra[1], Manish Kumar Tiwari[1], and Tran Gia Phong[1]

National University of Singapore, Singapore

Abstract. During the current global Covid-19 pandemic, digital contact tracing plays an irreplaceable part in limiting the spread of the disease, in particular the digital contact tracing protocols that utilize Bluetooth. As users' information (i.e. list of contacts made) is greatly involved, privacy has become a major concern in designing an effective protocol. As the result, various privacy-preserving contact tracing protocols have been proposed, generally involving users sharing anonymous IDs to record contacts. Motivated by this, in this paper, we will discuss the security of these protocols. More specifically, we provide a more formal treatment to the confidentiality and the integrity of the schemes, as well as having the DP-3T and BlueTrace protocols as case studies.

Keywords: privacy-preserving · contact tracing · proximity tracing · Bluetooth.

1 Introduction

Contact tracing has been playing an important role in limiting the spread of infectious diseases, which is further emphasized by the rise of the Covid-19 global pandemic since around the start of 2020. However, during the pandemic, manual contact tracing processes have become less effective as they are too time-consuming and cannot catch up with the spreading rate. As a result, this led to the development of digital contact tracing. It is a contact tracing method that relies mostly on people's portable devices such as smartphones.

Various technologies can be utilized for digital contact tracing. These include GPS, Bluetooth, or using QR codes. Using GPF for location tracking has been less accurate and has privacy concerning issues. While the use of QR codes is often limited to indoor buildings and venues, leaving out roads and public transport vehicles. On the other hand, Bluetooth proximity tracing is more accurate and has fewer privacy issues.

Bluetooth is used in proximity tracing for users to broadcast and exchange their anonymized IDs with each other when a contact is made. In the event of a positive diagnosis, the contact log is used to determine if any other user is at risk. There is also a central server, and depending on its role, we can divide

the contact tracing protocols into two classes, which are centralized and decentralized. Loosely speaking, in centralized proximity tracing, users' contact logs are reported back to a central server regularly, where the exposure risk is then computed and the at-risk users are notified back. On the other hand, in the decentralized setting, users' log is kept locally. In case of a positive diagnosis, only the anonymized ID is sent to the server, which is then broadcast to other users' devices. Exposure risk is then computed locally to determine if the user is at risk. Some examples of a centralized proximity tracing protocol are PEPP-PT [1] and BlueTrace [3], and some decentralized ones are DP-3T [4] and GAEN [2].

Contribution: There is a need for a formal treatment rather than a heuristic approach to the security of different contact tracing protocols. Therefore, in this paper, we formally define various security aspects of digital contact tracing protocols using Bluetooth, more specifically the confidentiality and integrity. Next, we pick the protocols DP-3T and BlueTrace as a representative for centralized and decentralized protocols, respectively, as case studies to discuss the security we defined. Finally, we present our conclusion regarding the security of the current digital contact tracing systems.

2 Formalization of Bluetooth Technology for Proximity Tracing

In this paper, we focus on contact tracing protocols that utilize Bluetooth technology for proximity tracing.

There are two types of entities within each protocol, which are multiple clients and a single server. We can view each protocol as having two concurrent processes:

1. Proximity tracing (client-client communication): for every client C_0 and C_1 that come into close contact:
   - For $i \in \{0, 1\}$, C_i generates an anonymous identification t_i and optionally stores information related to how t_i was generated.
   - C_0 sends t_0 to C_1, C_1 sends t_1 to C_0.
   - For $i \in \{0, 1\}$, C_i stores t_{1-i}.
2. Exposure computation and notification (client-server communication): this process only occurs when a client has a positive diagnosis, and varies between different protocols. Let C be the positive client, S be the server, and T be the time window the client is considered as contagious. The processes for the different settings are as follows:
   - Centralized setting: C sends S contact log within T (i.e. the received tokens), from which S computes the set $\mathcal{C}$ of at risk clients, then S notifies C' for all $C' \in \mathcal{C}$.
   - Decentralized setting: C send S all tokens generated (i.e. the sent tokens) or the corresponding seeds during T. S forwards the information to every other client C', and C' then computes whether the user is at risk.

3 Formalization of Security Aspects

Confidentiality and Integrity are defined with the two experiments below.

3.1 Confidentiality

Definition 1 (Adaptive chosen-token indistinguishability experiment). *An adversary chooses two persons P_0 and P_1 at a given time T. The challenger chooses a uniform random bit b in $\{0, 1\}$. The person P_b will generate a token t and give it to the challenger. The challenger will send t to the adversary. Throughout the experiment, the adversary has access to an oracle such that for any person P' and time T' where either $P' \notin \{P_0, P_1\}$ or $T' \neq T$, the oracle will return a token t', that P' would generate at time T'. The adversary will try to guess t, if t is generated by P_0 or P_1 output b'. The adversary wins if $b = b'$.*

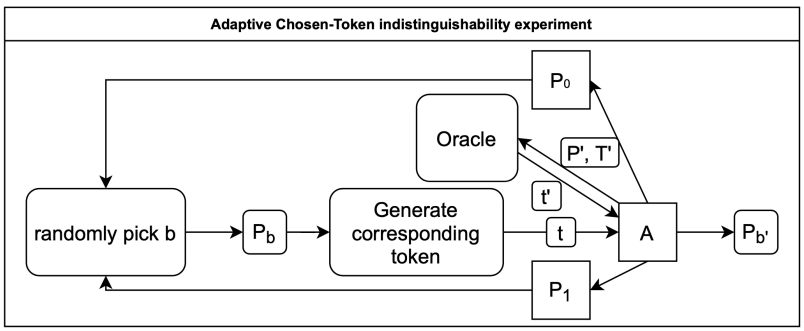


Fig. 1: Adaptive chosen-token indistinguishability experiment

A token is confidential if for any PPT $\mathcal{A}$, there is a negligible function negl such that the probability that $\mathcal{A}$ wins the experiment is less than $\frac{1}{2} + \text{negl(n)}$, where n is the security parameter.

3.2 Integrity

Definition 2 (Token duplication experiment).
A challenger gives a token t to the adversary. The adversary inputs information c into the contact tracing protocol, it then generates a token t', where t' $\neq$ t. If t' is a valid token, the system outputs 1 and the adversary has won the game. If the system deems the token is invalid, the system outputs 0 and the adversary has lost.

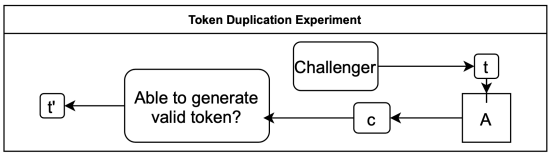


Fig. 2: Token duplication experiment

A token's integrity is secured iff for any PPT $\mathcal{A}$, there is a negligible function negl such that the probability that $\mathcal{A}$ wins the experiment is less than negl(n), where n is the security parameter.

4 Case study 1: DP-3T

4.1 Design Specification

Decentralized Privacy-Preserving Proximity Tracing (DP-3T) is a contact tracing protocol utilizing Bluetooth Low Energy technology. There are three variations of the protocol with the difference in the trade-off between security and computation cost. In order of increasing security as well as computation cost, they are: Low-cost design, Hybrid design, and Unlinkable design [4].

All versions of DP-3T use a token called ephemeral identifier EphID to do the handshake between devices in close proximity. The user's device generates and broadcasts an EphID regularly (e.g. every 15 minutes). We call each of these time intervals an epoch.
When two devices are within a close distance, each of the devices received and stores information related to the other's broadcasted EphID as well other information such as the signal attenuation. Figure 2 illustrates how the EphID from one device is broadcasted to another.

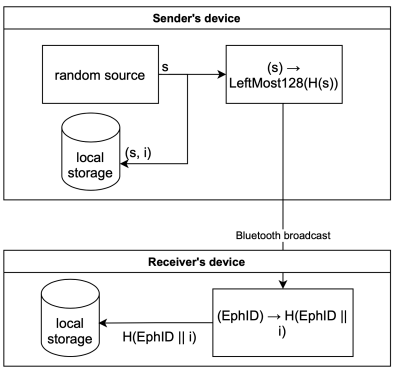


Fig. 3: Proximity tracing (Unlinkable design)

When a user gets a positive diagnosis, the information about the EphIDs (i.e. the seed and the epoch) generated within the time window that the user is contagious can be chosen to be sent to the server. Other users' devices regularly download this information from the server. The device determines if the user is at risk, by seeing if there is a match with any of the stored EphIDs previously received. Figure 3 illustrates this process.

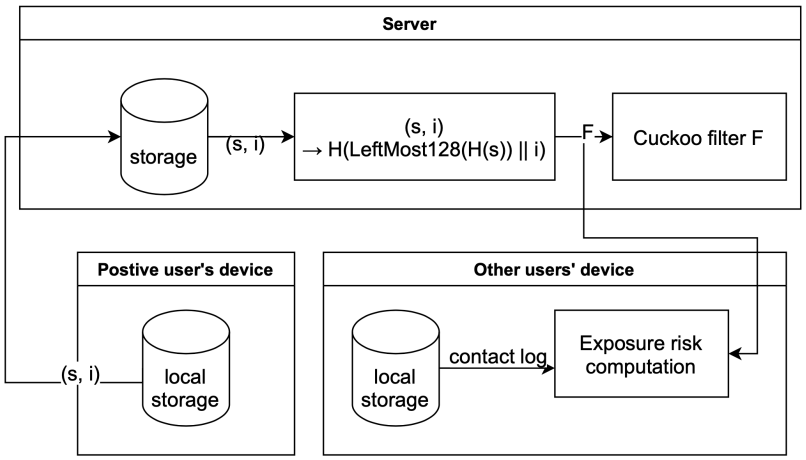


Fig. 4: Exposure computation and notification (Unlinkable design)

In this paper, we will only consider the unlinkable design. In the unlinkable design, to generate the EphID_i (EphID for the i-th epoch), the device first generates a random 32-byte seed s_i, and set $\text{EphID}_i = LEFTMOST128(\mathcal{H}(s_i))$

where $\mathcal{H}$ is a cryptographic hash function. These seeds with the corresponding epoch tag are also stored locally for 14 days. In case of a positive diagnosis, the device sends all pairs of (s, i) within the contagious time window of the positive user to the server. The server regularly (e.g. every 2 hours) creates a Cuckoo filter of $\mathcal{H}(LEFTMOST128((\mathcal{H}(s_i))||i)$ for every pairs of (s, i). This filter is then sent to other users' devices to help compute exposure risk locally.

4.2 Confidentiality Analysis

We will show that the unlinkable design is secure under Adaptive chosen-token attack.

Token: EphID
Adversary and Goal: The adversary is a tech-savvy user who can listen to the EphID broadcast by other users (either by listening honestly or using Bluetooth receivers). The goal is to distinguish who generated a particular EphID. With this ability, by setting up multiple Bluetooth receivers, the adversary can track the victim's location using the EphID generated.

In the setting of DP-3T (Unlinkable Design), the experiment is as follows: the adversary can choose 2 persons P_0 and P_1 at epoch i. The adversary also has access to an oracle that can compute the EphID generated by any person P at epoch i', as long as $(P_0, i) \neq (P, i')$ and $(P_1, i) \neq (P, i')$. Then at epoch i, the challenger randomly choose either P_0 or P_1 to generate the token t. Given t, if the adversary can guess who generated the token then they win.

Consider any PPT adversary $\mathcal{A}$ in the chosen-token attack indistinguishability experiment. First, we note that the seeds generated in each epoch are independent of each other. Since $\text{EphID}_i = LEFTMOST128(\mathcal{H}(s_i))$, which is a function with only the seed as the input, thus the EphIDs generated in each epoch are also independent of each other (i.e. the probability that an EphID is equal to certain value does not affect whether or not conditioning on previous epoches' EphID).

Hence, from $\mathcal{A}$'s perspective, all queries to the oracle are irrelevant (recall that $\mathcal{A}$ is not allowed to query (P_0, T) and (P_1, T)). This means that querying the oracle does not help $\mathcal{A}$ guess any better. However, the oracle is the only way $\mathcal{A}$ can differentiate between P_0 and P_1. Hence, $\mathcal{A}$ cannot do any better than truly random guessing, with the chance of winning is exactly $\frac{1}{2}$.

4.3 Integrity Analysis

The integrity of a system is broken when an adversary can send a valid token into a system. Before we begin our analysis, we shall define the terminologies below based on the context of DP-3T's system.
Token: $\mathcal{H}(\text{EphID}_i||i)$

Adversary and Goal: A non-positive Covid-19 user who wants the victim to receive an alert for close contact.

Case 1 (Trivial)

When a user is diagnosed with Covid-19, they can instruct their device to upload the EphID_is corresponding with their contagious period. The phone then uploads (i, seed_i) to the cuckoo filter as seen in the above Figure.

This is a form of vulnerability if there is no authority to check that a user who has uploaded their EphID_i are truly Covid-19 positive.

Thus an adversary can achieve the goal of sending a false alarm by trivially by:
1. Coming into contact with the victim
2. Sending the EphID_i corresponding to that period.

Case 2: Presence of Covid-19 Check
Assuming that the system only allows users to upload their EphID_i if they have a Covid-19 positive diagnosis.

An adversary can attack, by sending a valid (i, seed_i) to the Cuckoo filter and trick the victim's phone into downloading it. The following analysis will discuss the method above, with the assumption that the adversary knows what are the EphID, EphID_i where $i \in \{0, k\}$ are the ranges of EphID, present in the victim's phone.

To cause a false alarm, the adversary has to send a information c = (i, seed_i) to Cuckoo Filter, that produces a hash that coincides with token $\mathcal{H}(\text{EphID}_i||i)$ where $i \in \{0, k\}$.

Assume that the hash function $\mathcal{H}$ is collision-resistant and the adversary succeeds. If the adversary succeeds it means that the adversary managed to find a (i, seed_i). This will contradict the fact that $\mathcal{H}$ is collision-resistant. Hence if $\mathcal{H}$ is collision resistance, the adversary will be able to succeed in getting Cuckoo Filter to generate a valid token t' with probability less than negl(n). The system is thus secure, ensuring the integrity of the notifications.

5 Case study 2: BlueTrace (TraceTogether)

Design Specification

BlueTrace is the protocol used by TraceTogether to carry out contact tracing. It logs Bluetooth encounters between participating devices to facilitate contact tracing while protecting uses' privacy. User registration requires only a phone

number which is sent to the central server.

The server generates a random UserID and stores the UserID and phone number. Every 15 minutes, the server generates temporary IDs for all users and sends these TempIDs to the users' devices. These are stored locally and the server maintains a record of mapping of TempID to UserID.

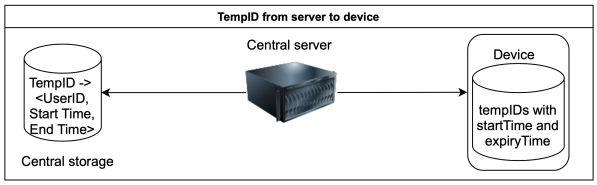


Fig. 5: TempID distribution

When two devices come in close contact with each other, they exchange their temporary IDs (TempID) with each other. The TempIDs of other users are stored locally on the device.

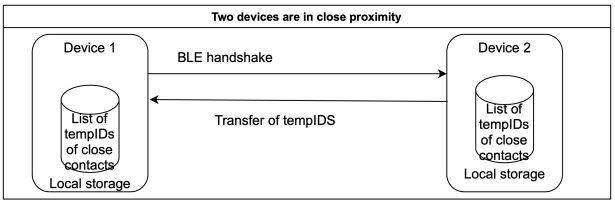


Fig. 6: Two Devices in close proximity

This data is only sent to the health authorities in the case when a user is confirmed to have tested positive for COVID-19.

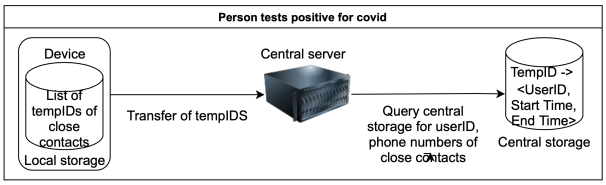


Fig. 7: Positive Case

A TempID comprises of UserID, start time, expiry time, which is encrypted with AES-256-GCM and is then Base64 encoded. Only the health authorities have the key to decrypt these TempIDs. Each TempID is generated with a random IV. The TempID includes the IV and an Auth Tag both generated from AES-256-GCM encryption scheme.

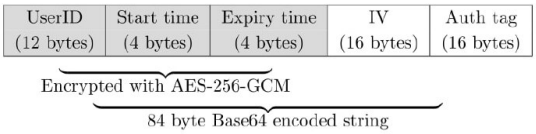


Fig. 8: TempID generation (taken from [3])

BlueTrace also makes use of Bluetooth Low Energy (BLE). Devices communicate with each other using this BLE protocol. Devices can take on peripheral or central roles. Peripheral devices advertise services, while central devices scan for Peripherals' advertisements to connect to their services. The data is exchanged in each "handshake" between a Peripheral and a Central. This message consists of 5 main fields, the TempID, device model(for calibration of distance estimates), organization code, BlueTrace version, and the Received Signal Strength Indicator. This is the least amount of information that is required to be exchanged to carry out effective contact tracing. All the fields except TempID do not carry any private information. The TempID, however, is encrypted using AES-256-GCM and thus, is computationally hard to decrypt without knowing the key.

5.1 Confidentiality Analysis

Confidential Token: TempID
Adversary and Goal: To determine the UserID of the person to whom the TempID belongs.

The adversary can set up Bluetooth receivers across the city, which log the TempIDs of users who come into contact. This log can now tell the adversary the identity and location of the users. The adversary can misuse this information in other means too.

We will show that the BlueTrace design is secure under adaptive chosen-token attack: Consider any PPT adversary $\mathcal{A}$ in the adaptive chosen-token indistinguishability experiment. Given 2 persons P_0 and P_1. The challenger chooses a uniform bit in 0,1. If b = 0, P_0 is choses as P_b, P_1 is chosen otherwise. Person P_b generates a TempID, t to the adversary. The adversary now has to determine whether t belongs to P_0 or P_1.

We also assume that $\mathcal{A}$ has access to an oracle that returns a TempID, for any person P $\notin \{P_0, P_1\}$ and time T. We first note that the encryption is done using a random 16 byte IV and a key unknown to $\mathcal{A}$. $\mathcal{A}$ can see AES-256-GCM(UserID, startTime, expiryTime) || IV || Auth Tag.

Using the oracle, $\mathcal{A}$ can obtain pairs of corresponding pairs of (UserID, time) and TempID. If $\mathcal{A}$ is able to query the oracle for P_0 or P_1, $\mathcal{A}$ can determine the UserID with probability 1. Since $\mathcal{A}$ is not allowed to query for P_0 or P_1, the problem comes down to executing a known-plaintext attack on an AES-256-GCM cipher. To figure out the key $\mathcal{A}$ has to try all 2^{256} sets of keys and find the one which matches the known pair. This is computationally infeasible. Hence, $\mathcal{A}$ has no way to differentiate who TempID, t belongs to. $\mathcal{A}$ cannot do any better than random guessing, where the chance of winning is exactly $\frac{1}{2}$.

Therefore, the BlueTrace token is confidential with regards to our definitions described in Section 3.

5.2 Integrity Analysis

The integrity of a system is broken when an adversary is able to send a valid token into the system.
Token: TempID
Adversary and Goal: To find a valid TempID t' when given another TempID t, such that t $\neq$ t'.

The Auth Tag is used for integrity check. The integrity is broken if the adversary is able to come up with a 20 bytes block that hashes into Auth Tag of t.

Assuming that a collision resistant keyed hash function H was used in AES-256-GCM, the probability that an adversary is able to find a hash collision would be negligible. Given the token duplication game, the probability that an adversary is able to come up with another valid TempID t' is negligible. Thus if a collision resistant hash function was used, the integrity of the token is promised.

Assuming that a non collision resistant hash function was used, and an adversary was able to find c = 20 bytes that hashes into the Auth Tag of t. The UserID, Start time and Expiry time in that block might be equal to a valid UserID, Start time and Expiry time with probability calculated as follows:
The total number of 4-byte representations of any point in time within the last 1 month is at most $\leq 31 \times 2 \times 2 \times 23 \times 59 \times 59$
The total number of value 4 bytes can represent is $2^{4\times 8}$
Hence Pr[both start and end time are valid] $\leq \frac{31\times 2\times 2\times 23\times 59\times 59}{2^{32}}^2 = 5.3 \times 10^{-6}$

Given the token duplication game, the probability that an adversary is able to come up w another valid TempID t' is 5.3×10^{-6}. Thus the integrity of a token can be compromised with a small probability if a non-collision resistant hash was used.

The hash function H used in AES-256-GCM is unlikely to have been a non-collision resistant hash function for the same key.

Here, we have analysed even if a non-collision resistant hash is used, the probability that the adversary is able to successfully generate a valid TempID is very low. Given, the actual hash function is collision resistant, the actual probability that the adversary can win is negligible. Therefore, BlueTrace is highly likely able to ensure integrity of the tokens.

6 Conclusion

The paper analyses confidentiality and integrity in the systems of DP-3T and BlueTrace. DP-3T uses Hash function for encryption while BlueTrace uses AES-256-GCM, which also contains a hash function. We have studied the confidentiality of both systems by playing a adapted chosen token attack system. DP-3T would have failed for a chosen token attack system given that it is deterministic while BlueTrace will not as it uses a random IV. The integrity of both systems are secured if collision resistant hash functions are used. All in all, both systems are said to be well-implemented and do not fall vulnerable in our experiments.

References

1. Pan-european privacy-preserving proximity tracing high-level overview. https://github.com/pepp-pt/pepp-pt-documentation/blob/master/PEPP-PT-high-level-overview.pdf. Accessed: 2021-11-01.
2. Privacy preserving contact tracing. https://covid19.apple.com/contacttracing. Accessed: 2021-11-01.
3. Jason Bay, Joel Kek, Alvin Tan, and Chai Sheng Hau. Bluetrace: A privacy-preserving protocol for community-driven contact tracing across borders. 2020.
4. Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Čapkun, David Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel Smart, Aysajan Abidin, Seda Gürses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain

Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. Decentralized privacy-preserving proximity tracing, 2020.

What makes a healthcare database anonymous? A security formulation of anonymity

Chan Shun Jie[1 [A0184044J]], Khoong Wei Kit[1 [A0149859E]], Mu Changrui[1 [A0225764Y]], and Tan Kai Li Catherine[1 [A0188596A]]
(Names in Alphabetical Order)
Supervised By Prof. Hugh Anderson[1]

National University Of Singapore, 21 Lower Kent Ridge Rd, 119077, Singapore

Abstract. This paper focuses on the use of databases in the healthcare industry, where patient data is often queried to aid the diagnosis and treatment of patients by doctors and nurses. During this process, privacy concerns exist where bad actors in the system might use personally identifiable information (PII) of the patient for malicious activities. Can the privacy of patients be maintained (i.e. remain anonymous) after selected information from their profile is accessed? Our group aims to propose the properties of anonymity and query schemes that preserve anonymity in a database.

Keywords: Anonymity · Database · Privacy.

1 Introduction

1.1 Background

Organizations often release datasets to the public in statistical publications, which contain information that was perceived to be private i.e. it does not directly identify individuals. For example, a set of disclosed attributes about an individual could be {zip code, birth date, gender}. However, attackers can perform re-identification by matching these attributes to other databases which contain directly identifiable information. For example, a 2nd database may have a set of attributes {name, zip code, birth date, gender, ... }. The attacker matches {zip code, birth date, gender} to get the name of the corresponding individual. These three attributes are sufficient to identify 87% of individuals in the US![1]

This leads to a severe problem where attackers with access to external databases can invade the privacy of individuals by performing cross-matching of attributes among these databases and thus identifying them. In this paper, we will introduce security schemes that can prevent the aforementioned re-identification attacks and prove their security. However, it is also necessary to introduce several formal definitions to model the system.

1.2 Properties of the System and Objectives

In this system, we assume that there exist multiple databases. A database is represented by a table $T = \{t_1, t_2, ...t_n\}$ where t_i corresponds to the i-th record in the table out of n total records. Suppose that we T is the database that we want to anonymize and protect against re-identification. T undergoes an anonymization process (k-anonymity / l-diversity / t-closeness), and is transformed to an anonymized table T*, which an adversary can query.

Each record in T and T* contain attributes $A = \{a_1, a_2, ..., a_m\}$ form number of attributes (i.e. age, zip code, weight, etc). Attributes can be classified as non-sensitive or sensitive attributes. Non-sensitive attributes are considered quasi-identifiers if they can be used to uniquely identify at least 1 individual from the database. For simplicity of discussion, let each individual be identified by quasi-identifiers Q and sensitive attributes S, such that a single record $t \in T$ is identified by t[Q] and possesses the sensitive attributes t[S].

With this system, the goal of the anonymization process is as follows. Given a database T, and external databases $D = \{d_1, d_2, ...\}$ (i.e. the adversary's background information), an adversary's posterior probability of identifying an individual's sensitive information, $t[S] \in T$, based on the known identifiers $t[Q] \in T$, should ideally be equal to the prior probability of identifying t[S]. This is synonymous with the uninformative principle [2], which states the published table should provide the adversary with little additional information beyond the background knowledge. In other words, there should not be a large difference between the prior and posterior beliefs. Relating to Perfect Secrecy and indistinguishability, a more formal definition of the uninformative principle is as such:

Definition 1 (Indistinguishable Privacy). *Given knowledge of some external databases D, a guess, s, made by the adversary of an individual's sensitive attribute, t[S], then given an operation which transforms the original table T to the anonymized table T*:*

$$|Pr(t[S] = s|D) - Pr(t[S] = s|D \cup T^*)| \leq \epsilon$$

where $\epsilon \geq 0$.

Ideally, we want $\epsilon = 0$ to imply negligible information leakage to the attacker by T*. To achieve a small ϵ, there is a need to reduce the information disclosed by the quasi-identifiers Q in T. This can be done by either generalizing Q or removing attributes completely from Q. One such security scheme is known as K-anonymity.

However, as it will be shown in later discussions, it is difficult to achieve a perfect anonymization scheme that has practical use. Firstly, existing practical security schemes do not scale exponentially in difficulty for an attacker, and an attacker can identify an individual in polynomial time. That is, given n trials by an attacker, $\epsilon \neq negl(n)$ and $\epsilon = 1/PPT(n)$. Secondly, a perfect security scheme

can guarantee anonymity based on definition 1, but it has little practical usage if non-malicious users want to query useful results from it. Thus, there will exist a trade-off between usability and anonymity in the system.

1.3 Privacy Threats

In addition to indistinguishable privacy, we aim to protect the database T from the following privacy threats, which are defined as follows:

Membership Disclosure - an adversary is able to determine whether an individual's data is in the table.

Attribute Disclosure - an adversary is able to infer the sensitive attributes of a given individual.

Identity Disclosure - an adversary is able to link an individual to a specific data entry. This usually entails legal consequences for the author or publisher of the database.

2 k-Anonymity

k-anonymity is defined as having at least k individuals in the anonymized dataset that share any set of quasi-identifiers. It means that every individual in the dataset can "hide" himself among a group of size k and above, such that their non-sensitive attributes are the same.

2.1 Formal Definition of k-anonymity [3]

Definition 2. *A published table T^* is k-anonymous if for every quasi-identifier q in T^*, there exist at least k records in T^* such that the quasi-identifiers are the same:*

$$\forall q \in Q \, in \, T^*, t_1[q] = t_2[q] = \ldots = t_n[q], \, where \, n \geq k. \tag{1}$$

2.2 How k-Anonymity is Achieved

k-anonymity is achieved through generalization and suppression. Generalization is the process of replacing specific quasi-identifiers with more general values until the dataset has k identical values. For instance, instead of revealing the specific ages of the individuals, the ages can be grouped into a broader age group so that a larger number of people will classify under the same group. On the other hand, suppression is the process of replacing the data in one or more quasi-identifiers with an asterisk, so that none of the values in those columns will be revealed.

2.3 Problems with k-Anonymity

While k-anonymity seems promising, there are problems with the anonymized dataset. First, it is susceptible to the homogeneity attack, which happens when within a set of k people with the same quasi-identifier, the sensitive values are identical. For example, in figure xxx, within the quasi-identifier "Age $\geq$ 50, Salary = \$0", all of them are diagnosed with COVID-19 and have the sensitive attribute "COVID-19 = Positive". Then, if an attacker knows that person A is in this dataset and has the above-stated quasi-identifier, the attacker can immediately know that person A has COVID-19, without knowing which row in the dataset corresponds to person A exactly. Second, it is susceptible to background knowledge attacks. This attack occurs when there is an association between the quasi-identifiers and the sensitive attributes such that the set of possible values for the sensitive attribute is decreased. For example, in figure xxx, within the quasi-identifier "Age = 30-50, Salary = \$0-\$4000", there are only 33% of individuals diagnosed with COVID-19. If an attacker knows that person B has the above quasi-identifiers and knows that person B is 50 years old, he can infer that there is a higher chance that person B has COVID-19 if it is known that people of an older age have a higher chance of getting COVID-19 due to their weaker immune system.

2.4 Properties of k-anonymity

Given that database table, T is transformed to the published table T^* using k-anonymity, and T and T^* both consist of n records, let membership disclosure, sensitive attribute disclosure, and identity disclosure be represented be m, s and i respectively.

$$Pr(m \mid T^*) \geq \frac{K}{n} \tag{2}$$

$$0 < Pr(s \mid T^*) \leq 1 \tag{3}$$

$$Pr(i \mid T^*) \leq \frac{1}{K} \tag{4}$$

In the worst case scenario,

$$|Pr(m \mid T) – Pr(m \mid T^*)| = |1 – \frac{K}{n}| \tag{5}$$

$$|Pr(s \mid T) – Pr(s \mid T^*)| = |1 – 1| = 0 \tag{6}$$

$$|Pr(i \mid T) – Pr(i \mid T^*)| = |1 – \frac{1}{K}| \tag{7}$$

Ideally, we want the differences to be as large as possible. However, the probability of sensitive attribute disclosure is still absolute in the worst-case scenario. With these problems, l-diversity is a more powerful privacy definition on top of k-anonymity that lowers the success rates of the attacks mentioned above by increasing the diversity of the dataset.

3 l-Diversity

The idea behind l-diversity is that to counter the homogeneity attack mentioned above, within each quasi-identifier, there should be "diversity" in the sensitive attributes so that even if the attacker knows an individual's quasi-identifier, there will be a low probability of getting the values of the sensitive attributes correct. l-diversity is defined as having at least l different values for the sensitive attributes for each quasi-identifier group. As such, even if the attacker knows the quasi-identifier of an individual, the probability of obtaining the correct value for the sensitive attribute is lower as there would be at least l possible values for the sensitive attributes. If the attacker has background knowledge, he needs to have (l-1) useful pieces of background knowledge in order to confirm the sensitive attribute of an individual.

3.1 Formal Definition of l-diversity [2]

Definition 3. *A published table T^* is l-diverse if for every quasi-identifier q in T^*, there exists at least l different sensitive attributes $t[S]$:*

$$\forall q \in Q \text{ in } T^*, t_1[q] = t_2[q] = \ldots = t_n[q] \text{ and } t_1[s] \neq t_2[s] \neq \ldots \neq t_m[s], \\ \text{where } n \geq k \text{ and } m \geq l$$

3.2 Problems with l-diversity

However, with l-diversity together with k-anonymity, the anonymized database is still susceptible to attacks. First, the skewness attack takes advantage of the difference in distributions of sensitive attributes within a quasi-identifier as compared to the whole dataset. An example would be as follows: In figure xxx, the probability of getting diagnosed with COVID-19 in the full database is 60%. However, there is a quasi-identifier such as "Age $\geq$ 50, Salary = \$0-\$8000" that has 75% of the individuals having COVID-19. Thus, an attacker knowing that an individual belongs to the above-stated quasi-identifier will now learn that the individual has a high probability of having COVID-19. The posterior probability is 75%, a higher value than 60%.

Second, the similarity attack takes advantage of semantically similar sensitive attributes within a quasi-identifier group. For example, the sensitive attribute in a dataset might be blood pressure levels. If the attacker knows that person C belongs to a quasi-identifier group that has all values of blood pressure between 120 mm Hg and 150 mm Hg, then the attacker can infer that person C has high blood pressure even though he does not know the exact value of person C's blood pressure

3.3 Properties of *l*-diversity

Given that database table T is transformed to published table T^* using l-diversity, and T and T^* both consist of n records, let membership disclosure,

sensitive attribute disclosure and identity disclosure be represented be m, s and i respectively.

$$Pr(m \mid T^*) \geq \frac{k}{n} \tag{8}$$

$$\frac{1}{l} < Pr(s \mid T^*) \leq \tag{9}$$

$$Pr(i \mid T^*) \leq \frac{1}{k} \tag{10}$$

In the worst case scenario,

$$|Pr(m \mid T) – Pr(m \mid T^*)| = |1 – \frac{k}{n}| \tag{11}$$

$$|Pr(s \mid T) – Pr(s \mid T^*)| = |1 – (1 - \frac{l-1}{k})| = \frac{l-1}{k} \tag{12}$$

$$|Pr(i \mid T) – Pr(i \mid T^*)| = |1 – \frac{1}{k}| \tag{13}$$

Ideally, we want the differences to be as large as possible.

4 t-Closeness

t-closeness is a novel, further refinement of the l-diversity property to prevent the aforementioned Skewness and Similarity Attacks. An equivalence class of a table is a set of data entries that is indistinguishable from one another through the quasi-identifier attributes. t-closeness reduces the granularity of every equivalence class and their sensitive values by ensuring that the distribution of these sensitive attributes in an equivalence class is close to that of the entire database by t. With t-closeness, this increases the difficulty of an adversary distinguishing an individual's information from the groups of data based on the discrepancies between the groups' attribute distribution and the published table's distribution.

4.1 Formal Definition of t-closeness [4]

Definition 4. *A group of data A derived from the published table T^* is said to have t-closeness if the distance between the distribution of the attributes in A and the distribution of the attributes in T^* is smaller or equal to the threshold t. If all groups of data $(A_1, ..., A_n)$ in T^* have t-closeness, then T^* is also said to have t-closeness.*

With t-closeness, we can modify the formal definition of the prior and posterior beliefs of an adversary as follows:

– P denotes the distribution of the sensitive attribute values in the table

- Q denotes the global distribution of the sensitive attribute values in the table
- B_0 denotes the prior belief (before looking at the table)
- B_1 denotes the belief after knowledge of Q
- B_2 denotes the belief after knowledge of Q and P

The difference between B_0 and B_1 is hard to control as Q is considered publicly-accessible information. As such, t-closeness aims to limit the difference between B_1 and B_2. This can be accomplished by limiting the difference between P and Q such that the adversary does not gain any significant knowledge from B_1 to B_2. Mathematically, closeness between P and Q implies closeness between B_1 and B_2.

However, if P and Q are too close, this would decrease the correlation between sensitive attributes and quasi identifier attributes, restricting the amount of useful information for research or statistical purposes. On the other hand, if P and Q are not close enough, the adversary is able to gain significant knowledge which results in attribute disclosure. As such, there is a need to measure the distance between these two distributions to determine their closeness.

4.2 Calculation of t

The Earth Mover's Distance (EMD) is introduced to calculate the closeness between P and Q with the semantic closeness between any two attributes taken into consideration. EMD calculates the minimal amount of work required to transform one distribution to another by moving distribution mass between each other. In this context, the EMD formula can be formally defined using P and Q where $P = (p_1, p_2, ..., p_m)$, $Q = (q_1, q_2, ..., q_m)$. We let d_{ij} be the ground distance between element p_i and q_j and aim to find f_{ij} which is the flow of mass from P_i to Q_j that minimises the overall work.

$$D[P, Q] = WORK(P, Q, F) = \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} fij$$

The exact calculation of EMD depends on whether the data type is numerical or categorical. For numerical attributes, let $r_i = p_i - q_i, (i = 1, 2, ..., m)$ the distance between P and Q can be calculated through the Numerical Distance formula:

$$D[P, Q] = \frac{1}{m-1}(|r_1| + |r_1 + r_2| + ... + |r_1 + r_2 + ...r_{m-1}|) = \frac{1}{m-1} \sum_{i=1}^{i=m} \left| \sum_{j=i}^{j=1} r_j \right|$$

For categorical attributes, two formulae can be used to calculate its distance depending on how the attribute is grouped, namely the Equal Distance and Hierarchical Distance formulae.

The Equal Distance formula considers the distance between two distinct attributes as 1.

$$D[P,Q] = \frac{1}{2}\sum_{i=1}^{m}|p_i - q_i| = \sum_{p_i \geq q_i}(p_i - q_i) = -\sum_{p_i < q_i}(p_i - q_i)$$

The Hierarchical Distance formula takes into account categorical attributes with parent/child groupings similar to that of a tree structure. An example of such a structure can be found in the Appendix, Fig 2. For simplicity, the mathematical calculation has been omitted. The full details can be found in the paper written by Li et al.

4.3 Properties of *t*-closeness

As t-closeness can be applied to only a part of a database or to every equivalence class, it is important to note that it has the following two properties:

Theorem 1. ***Generalisation Property****: Given two generalisations A and B derived from a published table T^* where A is more general than B, if T^* uses B to satisfy t-closeness, then T^* can also use A to satisfy t-closeness.*

Proof. It can be observed that the equivalence classes in A are derived from the union of a set of equivalence classes in B. From the theorem, every equivalence class in B satisfies t-closeness which means that every equivalence class in A also satisfies t-closeness. Therefore, if T^* can satisfy t-closeness through B, it can also satisfy t-closeness through A.

Theorem 2. ***Subset Property****: Given a set of attributes C derived from a published table T^*, let D be any set of attributes where $D \subset C$. If T^* satisfies t-closeness using C, then T^* also satisfies t-closeness using D.*

Proof. It can be observed that every equivalence class relative to D is the union of a set of equivalence classes relative to C. Since every equivalence class relative to C satisfies t-closeness, this means that every equivalence class relative to D also satisfies t-closeness. Therefore, if T^* satisfies t-closeness using C, T^* also satisfies t-closeness using D.

As such, given a published table T^* with t-closeness and let the sensitive attribute disclosure be represented as s, the difference between the adversary's beliefs B_1 and B_2, as defined in Section 5.1, can be generalised as:

$$Pr(s \mid B_2) - Pr(s \mid B_1) \leq t \leq \text{negl}(n)$$

This means that an adversary who has information of P on top of Q does not gain any significant knowledge as compared to when they had information of Q only. The value of t can be determined by the author or publisher to ensure a balance between the degree of attribute disclosure and the comprehensiveness of the database while maintaining the negligence between the adversary's beliefs.

5 Differential Privacy: Membership Indistinguishability

Membership disclosure cannot be prevented by k-anonymity, l-diversity and t-closeness, because if we define two events:

- E_0: p's identifier fit in to some quasi-identifier group of the database
- E_1: p is in the database

then $Pr[E_1|E_0] >> Pr[E_1]$. For defencing membership disclosure, a game-based definition, namely differential privacy, is introduced in the model for preventing membership disclosure[5].

5.1 Differential privacy distinguishing game($\boldsymbol{Diff\text{-}Priv_{\mathcal{A},San}}$)

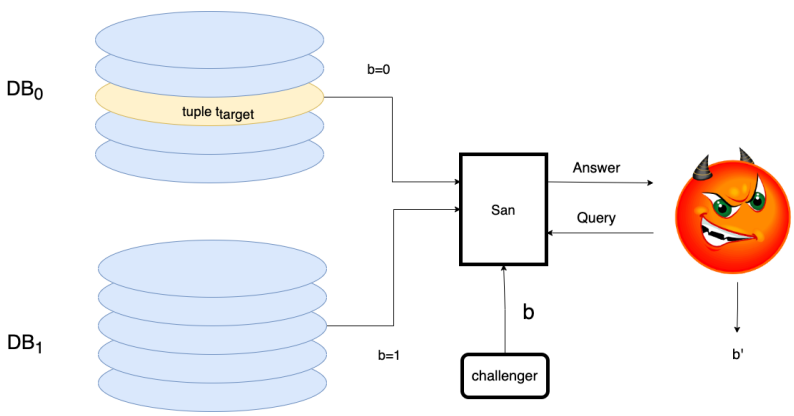


Fig. 1. Differential privacy distinguishing game

1. $\mathcal{A}$ selects t_{target}
2. Challenger generates two databases DB_0, DB_1, which are different in only one record($DB_0 \backslash DB_1 = t_{target}$) and $|DB_0| = p(n)$, where p is a polynomial;
3. Challenger generates a uniform bit $b \xleftarrow{R} \{0,1\}$, which is kept secret from all adversaries;
4. San is a function on DB, San is applied to database DB_b ;
5. Adversary $\mathcal{A}$ has multiple access to San for queries;
6. After all queries, $\mathcal{A}$ outputs a bit b';
7. $\mathcal{A}$ wins if b=b', denoted as $Diff\text{-}Priv_{\mathcal{A},San}(n) = 1$.

DB_0, DB_1 are said to be indistinguishable, or San is membership distinguishable, if $\forall \mathcal{A}, \exists$ a negligible function $negl$ such that $Pr[Diff\text{-}Priv_{\mathcal{A},San}(n) = 1] \leq \frac{1}{2} + negl(n)$.

Theorem 3. *If San is membership indistinguishable, then a database applied San can defend membership disclosure.*

Proof. This can be proved by reduction. If the database applied San is vulnerable to membership disclosure for certain adversary $\mathcal{A}$ efficiently, then a efficient adversary can be constructed to win $Diff\text{-}Priv_{\mathcal{A},San}$ by utilizing $\mathcal{A}$ to disclose the membership of t_{target} in DB_b and further wins $Diff\text{-}Priv_{\mathcal{A},San}$ efficiently.

5.2 Trade-off between privacy and usability

The sanitation function San takes one database and a bit and a query as input and the query answer as output. In theorem, the San function that achieves $Pr[Diff\text{-}Priv_{\mathcal{A},San}(n) = 1] \leq \frac{1}{2} + negl(n)$ is easy to construct. For example, San can be a random oracle, which will output random answer with response to adversary's query and $Pr[Diff\text{-}Priv_{\mathcal{A},San}(n) = 1] \leq \frac{1}{2}$ by the definition of random oracle[6]. However, construction in this way achieves high level of privacy but low level of usability.

Therefore, there is a trade-off between the usability and privacy in the design of San. In practice, San function may introduce generalization and suppression to achieve k-anonymity, l-diversity and t-closeness, which achieve certain polynomial level of privacy and also maintain high level of usability of the system. Moreover, San function can also be a statistical generator, which will do computation for users without releasing the raw data. Related techniques may include homomorphic encryption(HE), multi-party computation(MPC) and private set intersection(PSI) etc.[7][8].

6 Private Set Intersection

Healthcare entities usually store the data in form of medical analysis. Because the detailed medical data(e.g. x-ray image, electrocardiogram) takes a huge amount of storage requirements, some hospitals may require distributed databases for storing all the data. Consider the scenario where different databases together store private data of a patient, privacy may be leaked if the join query in the distributed databases is not secure[8]. Moreover, there are scenarios where two hospitals cooperate and share the records of the common patients. In this case, each party in the protocol should only learn information of common users but nothing else. In these two healthcare scenarios, private set intersection(PSI) can be applied to achieve the security requirements[9].
Our proposed Diffie-Hellman-based PSI for data sharing between hospitals allows the efficient and oblivious intersection of two databases. In the protocols

setting, two parties, namely Alice and Bob, both hold a set of records. After the execution of the protocol, Alice can get the intersection between the two sets without learning any uncommon data and also preserve all her data from Bob:

Alice with p		Bob with p
$X = \{x_1, x_2, ..., x_n\}$		$Y = \{y_1, y_2, ..., y_n\}$
Choose a random number $a \in \mathbb{Z}_p$	$\{H(x_1)^a, H(x_2)^a, ..., H(x_n)^a\} \rightarrow$	
	$\leftarrow \{H(x_1)^{ab}, H(x_2)^{ab}, ..., H(x_n)^{ab}\}$	Choose a random number
	$\leftarrow \{H(y_1)^b, H(y_2)^b, ..., H(y_n)^b\}$	$b \in \mathbb{Z}_p$
$\{H(y_1)^{ba}, H(y_2)^{ba}, ..., H(y_n)^{ba}\}$		
$\cap$		
$\{H(x_1)^{ab}, H(x_2)^{ab}, ..., H(x_n)^{ab}\}$		
Alice Output		Bob Output
If $x_i = y_i$, then $H(x_i)^{ab} = H(y_i)^{ba}$		$\perp$
If $x_i \neq y_i$, then $H(x_i)^{ab} \neq H(y_i)^{ba}$		$\perp$

6.1 Proof of Accuracy

Let H be a collision resistant hash function, p is a big prime number and a,b are chosen randomly from $\mathbb{Z}_p$. Then $Pr[H(x_i)^{ab} = H(y_j)^{ba} | x_i = y_j] = 1$ (because $H^{ab}(\cdot)$ is deterministic)and $Pr[H(x_i)^{ab} = H(y_j)^{ba} | x_i \neq y_j] = 1 \leq negl(n)$ (by reduction to collission resistancy of H)

6.2 Proof of Privacy

Bob:

- Under the Assumption that H is one-way and RSA problem is hard, Bob will be hard to get the information of any x_i by observing $H(x_i)^a$;therefore, Bob will not even know the data Alice has in common.

Alice:

- Under the Assumption that H is one-way and RSA problem is hard, it's hard for Alice to get the information of any y_i by observing $H(y_i)^b$.

Eavesdropper:

- Under the assumption that H is one-way and RSA problem is hard, Eavesdropper will be hard to get the information of any x_i by observing $H(x_i)^a$ and $H(x_i)^{ab}$, and also hard to know y_i by observing $H(y_i)^b$.
- Under the assumption that discrete logarithm is hard, Eavesdropper will not obtain $H(x_i)^a$ by observing $H(x_i)^{ab}$, and it does not know what x_i is, so it will not known the data stored in Alice and Bob in common.
- Under the assumption that computational Deffie-Hellman problem is hard, by observing any $H(x_i)^a, H(y_i)^b$, it will be computationally infeasible for the adversary to generate legal $H(y_j)^{ab}$ by itself.
- If the a and b is chosen properly to achieve decisional Deffie-Hellman secure, then $H(x_i)^{ab}$ will be indistinguishable from $H(x_i)^{2*r}$, where r is a truly random number
- Adversary do not have value a,b; So given any message x, it is computational infeasible for adversary to compute $H(x)^{a,b}$ and figure out the membership of x in the two database, hence membership disclosure is prevented.

It's worth mentioning the hash function H obliterates the multiple relationships between messages, which is necessary here because the RSA-style encryption is multiplicatively homomorphic. Moreover, there are no duplicates in the two sets, and an adversary cannot distinguish by spatial and local relationship.

6.3 Attacks on the Protocol

The security of the protocol should rely on the assumption that all parties are honest but curious. However, this protocol is insecure under active DOS attack, where adversaries may drop some records or mix the sequence of tuples to prevent Alice fail to get the joint set.

7 Conclusion

In conclusion, we have introduced three threats model of an anonymous database, namely membership disclosure, attribute disclosure and identity disclosure. Sequentially, the paper discusses k-anonymity, l-diversity and t-closeness as expected property of the anonymous database for defending attribute disclosure and identity disclosure. A membership distinguishing game is introduced to evaluate the probability of membership disclosure and security of sanitation function, where the trade-off between the privacy and the usability of sanitation function is discussed. Under different scenarios, the sanitation function may variate to achieve the expected usage and privacy expectation. In practice, the most common sanitation functions apply generalization and suppression to achieve k-anonymity, l-diversity and t-closeness, but there are scenarios where raw data

are not released during the query, and the privacy of data relies on the query schemes. In detail, we introduced a Diffie-Hellman-based PSI that achieves oblivious join query and defends membership disclosure against passive adversaries. This raises the interesting construction of a combination of anonymous data set and privacy-preserved query schemes as a feasible solution of anonymous database systems, which is a topic for further research.

References

1. L. Sweeney, “Simple Demographics Often Identify People Uniquely”. Carnegie Mellon University, 30-Jun-2018
2. A. Machanavajjhala, J. Gehrke, D. Kifer and M. Venkitasubramaniam, ”L-diversity: privacy beyond k-anonymity,” 22nd International Conference on Data Engineering (ICDE’06), 2006, pp. 24-24, https://doi.org/10.1109/ICDE.2006.1.
3. L. Sweeney. k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, 10 (5), 2002; 557-570.
4. N. Li, T. Li and S. Venkatasubramanian, ”t-Closeness: Privacy Beyond k-Anonymity and l-Diversity,” 2007 IEEE 23rd International Conference on Data Engineering, 2007, pp. 106-115, https://doi.org/10.1109/ICDE.2007.367856.
5. A. Kosky, “Observational distinguishability of databases with Object Identity,” International Workshop on Database Programming Languages, 1995.
6. R. Canetti, O. Goldreich, and S. Halevi, “The random Oracle Methodology, revisited,” Journal of the ACM, vol. 51, no. 4, pp. 557–594, 2004.
7. D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, “From keys to databases—real-world applications of secure multi-party computation,” The Computer Journal, 2018.
8. T. Lepoint, S. Patel, M. Raykova, K. Seth, en N. Trieu, “Private Join and Compute from PIR with Default”. 2020.
9. T. Duong, D. H. Phan, en N. Trieu, “Catalic: Delegated PSI Cardinality with Applications to Contact Tracing”. 2020.

A Appendix

Table 1. Privacy Achivements

	Membership Disclosure	Attributes Disclosure	Identity Disclosure
k-anonymity	Vulnerable	Vulnerable	Defensible
l-diversity	Vulnerable	Semi-defensible	Defensible
t-closeness	Vulnerable	Defensible	Defensible
Membership Indistinguibility	Defensible	Defensible	Defensible

Table 2. Unprocessed Data

Age	**Salary**	**COVID-19**
15	$0	Negative
24	$4000	Negative
30	$3500	Positive
32	$4800	Positive
35	$5200	Negative
44	$6000	Positive
50	$8000	Negative
62	$0	Positive
76	$0	Positive
78	$0	Positive

Table 3. 3-Anonymous Data

Age	**Salary**	**COVID-19**
0-30	$0-$4000	Negative
0-30	$0-$4000	Negative
0-30	$0-$4000	Positive
30-50	$4000-$8000	Positive
30-50	$4000-$8000	Negative
30-50	$4000-$8000	Positive
30-50	$4000-$8000	Negative
> 50	$0	Positive
> 50	$0	Positive
> 50	$0	Positive

Table 4. 3-Anonymous, 2-Diverse Data

Age	**Salary**	**COVID-19**
0-30	$0-$4000	Negative
0-30	$0-$4000	Negative
0-30	$0-$4000	Positive
30-49	$4000-$8000	Positive
30-49	$4000-$8000	Negative
30-49	$4000-$8000	Positive
≥ 50	$0-$8000	Negative
≥ 50	$0-$8000	Positive
≥ 50	$0-$8000	Positive
≥ 50	$0-$8000	Positive

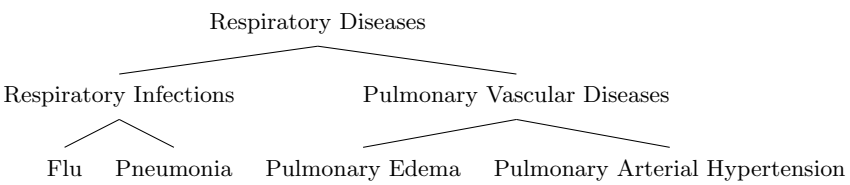


Fig. 2. Hierarchical Structure of Respiratory Diseases

Multi-Agent Consensus and Integrity in Autonomous Vehicles

TSAI, HSIAO-HAN[A0184742Y], Banerjee Aditya [A0200602E],
Tng Jun Wei[A0244043R], and Ho Wei Jun Sherman[A0201413A]
National University of Singapore

Abstract. Self-driving cars have risen in popularity over the years. As they become increasingly likely to be adopted by the public, there is a need to discuss and consider the security behind the self-driving system. Little had been done to discuss vulnerabilities and the corresponding security properties. In this paper, we had carefully considered the challenges self-driving cars will face and arrive at two security properties important for the self-driving cars system - multi-agent consensus and data integrity. Furthermore, we had derived game models in an attempt to show that the two security properties will ensure that the system is secure.

Keywords: Autonomous Vehicles, Blockchain, Cryptography

1 Introduction

Self-driving cars are a popular concept in recent years. Many companies have been developing self-driving cars and it is frequently a headline of news. For example, during the 2020 Tokyo Olympics, self-driving cars were used as a transportation method.

Although the use of the phrase “self-driving car” has seen a significant growth in popularity in the last decade, the concept itself is anything but new.

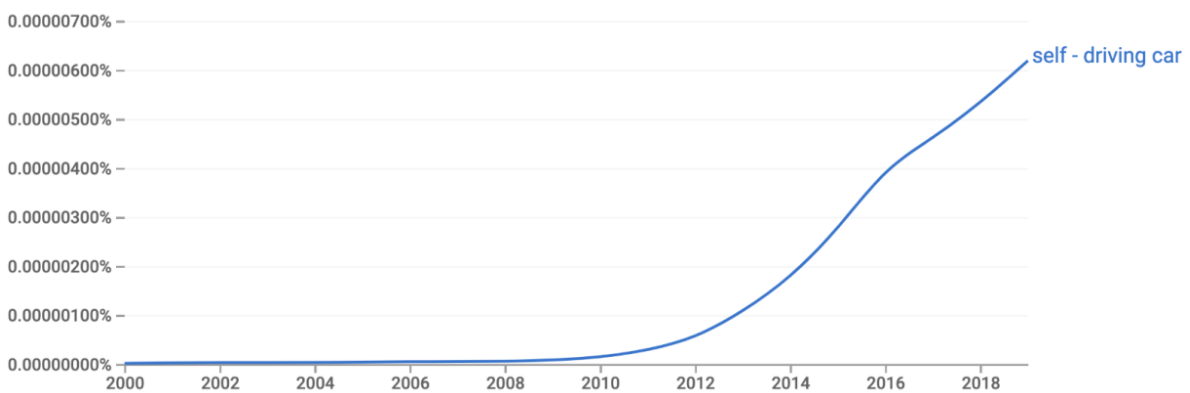


Fig. 1. Appearance of the phrase self-driving car in literature over time (Google, 2021)

The first “self-driving car” was unveiled by Norman Belle Geddes at GM’s exhibit at the 1939 World’s fair. A wire with current flowing through it was concealed under the road. Sensors at the front of the car that could detect the current, and steered left or right accordingly (Gringer, n.d.). One might argue that the car was not completely autonomous given the current had to be manipulated manually.

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Fig. 2. The different levels of autonomy for vehicles (Ondruš et al., 2020)

Fast–forward to today, vehicles have progressed beyond being self-driving, paving the way for a new generation of vehicles – autonomous vehicles. One key difference between these two classes of vehicles is that an autonomous vehicle is self-aware, and makes its own decisions, while a self-driving car can only make decisions pertaining to actually driving (Synopsys, n.d.). In the infographic above, self-driving cars would be categorized under level 3 or level 4, whereas autonomous cars would be categorized under level 5.

Autonomous cars rely on a number of components including LiDAR, Radar, Ultrasonic Sensors, Video Cameras, and a CPU among others. LiDAR is used to simulate the environment around the car in all directions using laser, ultraviolet, visible and infrared lights. The radar system is used to detect other vehicles and their speeds, obstacles and blind spots, among other applications. Ultrasonic sensors are used to detect objects in the vicinity of the vehicle. Video cameras are used to generate 3D images of the surrounding environment, in order to account for aspects of the environment that previously mentioned systems cannot comprehend, such as road signs. Data from the aforementioned sensors is processed by the CPU, which processes this data in real-time and makes decisions accordingly (Ondrus et. Al, 2020).

It may still be a number of years before fully autonomous vehicles can take the road, due to the advancements required in a number of different technologies implemented in such vehicles. In this paper, we focus on vehicle-to-vehicle communication. Currently, such communication relies on Dedicated Short-Range Communications (DSRC) technology. This allows vehicles to communicate with each other without the use of any external network infrastructure. The information communicated usually includes vehicle location and velocity, and is transmitted 10 times per second (Autotalks, n.d.). The main advantage of advancements in such technology is that vehicles could use sensor data from other vehicles along with its own to make better driving decisions.

Set in a world where all vehicles are fully autonomous, our paper focuses on the use of multi-agent vehicular communication to share information regarding vehicle behavior. As is the case with any form of network communication, such a system would be prone to attacks by malicious adversaries. In this paper, we aim to formalize security properties that would protect good actors against such attacks, and a defense mechanism that would uphold said properties.

2 Challenges faced by self-driving

Several security issues can arise when the control of a vehicle extends beyond its physical boundary, i.e. the movement and decision making of the vehicle are no longer dependent on a human agent, or an intelligent computer agent on-board the vehicle.

Firstly, malicious cars can influence the behaviors of other good cars by sending them cleverly-designed messages.

Here, we differentiate vehicles into two different types – malicious and good. Good cars are agents whose aim is to simply drive from point A to point B. The intentions of malicious cars are not to travel to a destination, but to cause unspecified side effects to the overall traffic.

Malicious cars will send various communication messages to good cars, which are capable of causing these problems.

1. They can send false sensor readings, or any other important vehicular metrics to other cars. For example, sending the "I am going to crash" signal may cause other vehicles to swerve and brake to avoid an imminent major collision. This could cause minor collisions instead.
2. They can send huge amounts of information-less messages, overwhelming the processing systems of good cars. These information-less messages are in essence, gibberish, yet they are considered legitimate and would be processed by the good cars. However, they are discarded upon further processing by the good cars. Large amounts of such messages can cause a slowdown in vehicular processing speed, or even stop the vehicle entirely.
3. They can send adversarial inputs to ML models onboard good cars, which can cause these cars to behave abnormally.

After good cars identify the above behavior, they can treat the car causing the problem to be a malicious car. If good cars can distinguish and exclude malicious cars from the communication network, then the first security issue could be mitigated to some extent. This can be achieved by allowing cars to propose keeping a record of behaviors, and more importantly, the behaviors exhibited by malicious cars. There needs to be a secure way of keeping these records, such that they are not easily forgeable and yet easily accessed.

Even if such behaviors can be tracked through a record, we need to ensure that the actual behaviors recorded are accurate. This maintains the overall integrity of the records.

In this paper, we seek to formalize and analyze these two security properties – (1) multi-agent consensus, (2) data integrity.

3 Security properties

1. Multi-agent consensus
   A group of agents should always accept a good proposal and reject a bad proposal.
2. Data integrity
   Assuming there are more than one proposal describing the same thing, if all good agents agree on the proposals, i.e. they believe that some values are plausible, then the accepted value will be the one closest to the ground truth.

4 Defense mechanism

To allow clear differentiation between good and malicious cars, the concept of "credibility" could be used. Malicious cars are likely to possess poor records of driving, e.g. sensor data and blacklisted data. Blacklisted data are data that the good cars consider as malicious (examples in Challenges). In order to verify if a car is malicious or not, a car (checker) can check a common database of records. This database is localized in each car and contains a compressed form of data sent by all cars in the traffic network. We assume that the checker can efficiently query the database for the necessary information and get an answer within an appropriate amount of time.
In that case, the only issue that antagonizes this method of verification is forgeability. As mentioned in the Challenges section, a malicious car can pretend to be a good car. This is done by changing its previous poor records, such that other cars will consider it to be good instead.
Hence, we need a way to store the records such that all the cars' databases agree with one another yet does not allow easy modification of the database.

We want to achieve this property by using blockchain. Blockchain has a characteristic of decentralization, which stores the data across peer-to-peer networks and does validation on each peer side. The peer-to-peer network maintains a huge amount of replication of the records. It also uses the proof-of-work concept. The time for challenger response is longer than the time for solution verification. Blockchain uses hash value as validation, and for the hash value, there is a certain requirement to achieve for example the first four bits of hash value need to be 1010. Using this method, it will have a huge computational power to get the correct hash value, and all the nodes will compete to calculate it for the next block. Even if one node tries to forge a block with malicious or false value, it still needs to have most of the other nodes' approval to pass verification. Since verification is much faster than the computation, unless the attacker has more than 51% of the total computational power, it cannot make changes to the records fast enough (more records would have been added since the last one). Thus, blockchain can maintain the stability of the records and still be able to achieve consensus.

5 Definition of our proposed system

In our system, each car has a record of behavior associated with it. This record utilises blockchain technology and is initialised upon registration. Each record in a car's blockchain will contain information about the car's behaviour in an interaction with another vehicle, the digital signature of the message, as well as the certificate signed using the other vehicle's private key.

In the event of an interaction between two vehicles, where one vehicle displays improper driving behaviour, the other vehicle will attempt to add this instance of behaviour to the blockchain of the vehicle displaying improper behaviour. In order for this instance to be accepted, the remaining vehicles must agree to the addition of the behaviour instance.

6 Assumptions

In the formulation of our game model, we made some simplifying assumptions pertaining to our proposed system.

- All self-driving cars are in level 5
- Behavior shouldn't change over the time
- All cars already have some records
- Has some method to distinguish bad behavior from good
- All cars on the road are self-driving cars
- All self-driving cars are using broadcasting to communicate.
- All self-driving cars are assigned with a certificate.
- Self-driving cars' behaviour records are stored in blockchain

- Attackers have less than or equal to 1/6 of total computational power in the self-driving car network.

7 Game Models and Analysis

7.1 Consensus

Game Model. Our game model involves the achievement of consensus among multiple agents. The game is comprised of four categories of players:

1. *Proposer*: This player aims to append an instance of behavior to the subject's vehicle.
2. *Subject*: The player whose record the proposer wishes to append an instance of behavior to.
3. *Conductor*: This player conducts the process of consensus achievement. They are tasked with selecting a set of validators and collating their votes with regards to the consensus.
4. *Validators*: A randomly chosen set of player vehicles, selected by the conductor. They each cast a vote with regards to whether the proposer should be allowed to append the aforementioned instance of behavior to the subject's vehicle.
5. *Idle*: Any player vehicles not involved in the process of achieving consensus.

Here we assume the Adversary A to be a set of users, they can be Proposers or Validators.

The proposal consensus is split into two scenarios:

1. The *Proposer* is not malicious. In this scenario, the adversary *A* wins/outputs 1 if the *Proposer* is unable to append the instance of behavior to the *Subject*'s record.
2. The *Proposer* is malicious. In this scenario, the adversary *A* wins/outputs 1 if the *Proposer* (in this case the adversary) is able to append the instance of behavior to the *Subject*'s record.

The game model design for $Consensus - Validation_{\pi,A}(n)$ is defined as below:

1. A random bit b is generated by a random generator with equal probability.
2. Random bit b determines the behavior of the *Proposer*. When b = 0, *Proposer* would form a malicious proposal (fake record). When b = 1, *Proposer* would form a good proposal.
3. The *Conductor* will choose a set of *Validators*, which is a subset of total users.
4. *Validators* have access to the *Oracle*, which can tell whether the proposal is malicious. A *Validator* can be either good or bad. A good *Validator* always rejects malicious proposals and accepts good proposals. A bad *Validator* rejects good proposals and accepts malicious ones.
5. The output of the final decision b' subject to majority of *Validators*' decision. b' = 0 means rejection of the proposal, b' = 1 means acceptance.
6. In the final step, there will be a check for b and b'. Adversary *A* succeeds when b is not equal to b', which means they are able to control the acceptance of a malicious proposal and rejection of a good proposal.

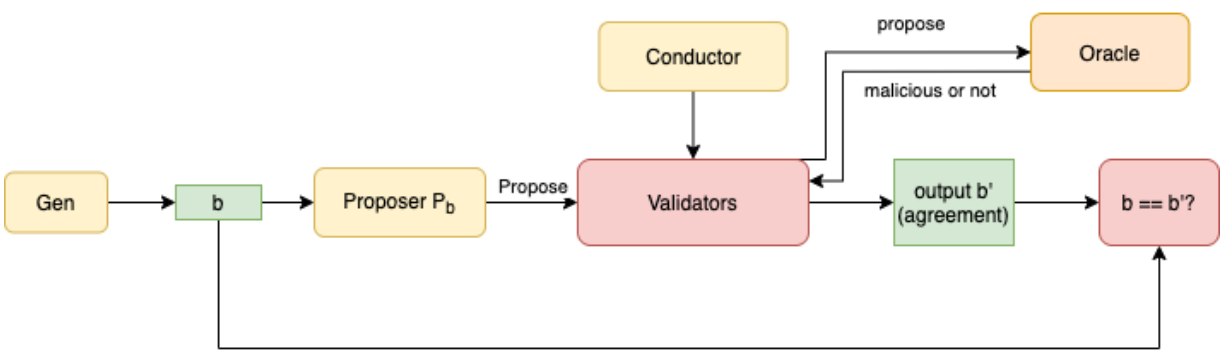


Fig. 3. Game Model for Consensus-Validation

Here, we propose that the scheme is *Consensus-Validatable* if and only if for any probabilistic polynomial time adversary, there exists a negligible function such that:

$$Pr[Consensus - Validation_{\pi,A}(n) = 0] \leq negl(n)$$

The decision of the system depends on the *Validators*, so the *Conductor*'s method of selection and the size of the *Validators* set would be key factors. One possible way is making the selection to be randomized and using the assumption of number if malicious users and selection size.

Model Analysis. We can prove that this system is *Consensus-Validatable* by the construction.

1. The *Conductor* has a bounded time for making a selection and would just randomly select the cars as *Validators*.
2. Since this is a randomized selection, we can treat it as a sampling, which the expected ratio of malicious *Validators* over all *Validators* will be approximate to the ratio of all malicious cars over total cars.
3. Assume the total car size is N. Using the previous established assumption, the malicious cars only have ⅙ N, and we select ½ N cars as *Validators*. To control a decision, the adversary *A* needs to have over half of the *Validators*, which needs to have at least ¼ N malicious *Validators*. Using (2), we can know that the expected malicious *Validators* are ½ * ⅙ = 1/12 N. Even in the worst-case scenario, which means all malicious cars are selected as *Validators*, the number of malicious *Validator*s is still less than ¼ N. No matter whether the proposal is good or malicious, the adversary *A* cannot control the decision of validation voting. Thus, $Pr[Consensus - Validation_{\pi,A}(n) = 0] \leq negl(n)$.

Our system has a strict assumption of the number of *Validators* to be selected and the number of malicious users. However, in the real situation, there should be other rules on the selection of *Conductor* and *Validator* to prevent a malicious proposal from being accepted. Another mechanism is giving a public reputation, which indicates the probability of a car/user to be selected as *Conductor* or *Validator*. If a malicious proposal has been accepted, the reputation of the *Conductor* and *Validators* would be deducted. Also, there will be rewards for correct validation. Every cars will be assigned with an initial public reputation. As more and more cars are added in the autonomous network, the probability of a malicious *Conductor* and *Validators* be selected will be significantly decrease.

Another characteristic of consensus that is not include in the game model is *Consensus-Unforgeable*. This characteristic is about after a proposal is accepted by the *Validators* and stores in the distributed nodes, it would have less than negligible probability to be modified. This characteristic is achieved by the proof-of-work concept. A block in the blockchain will contain the hash of the previous block. Modifying a block means the hash value of this block will be modified, which affects all the blocks appended to it. Thus, to successfully modify a block, multiple block values need to be recomputed. At the same time, the chain will become longer because there are other users appending to the original chain, which makes the number of blocks needing to be modified increase. Since the computational power of every user has no significant difference, the adversary cannot decrease the number of blocks he needs to modify. Our system is designed with blockchain, thus it is *Consensus-Unforgeable*.

7.2 Data Integrity

Game Model. For data integrity, we propose the following game model.

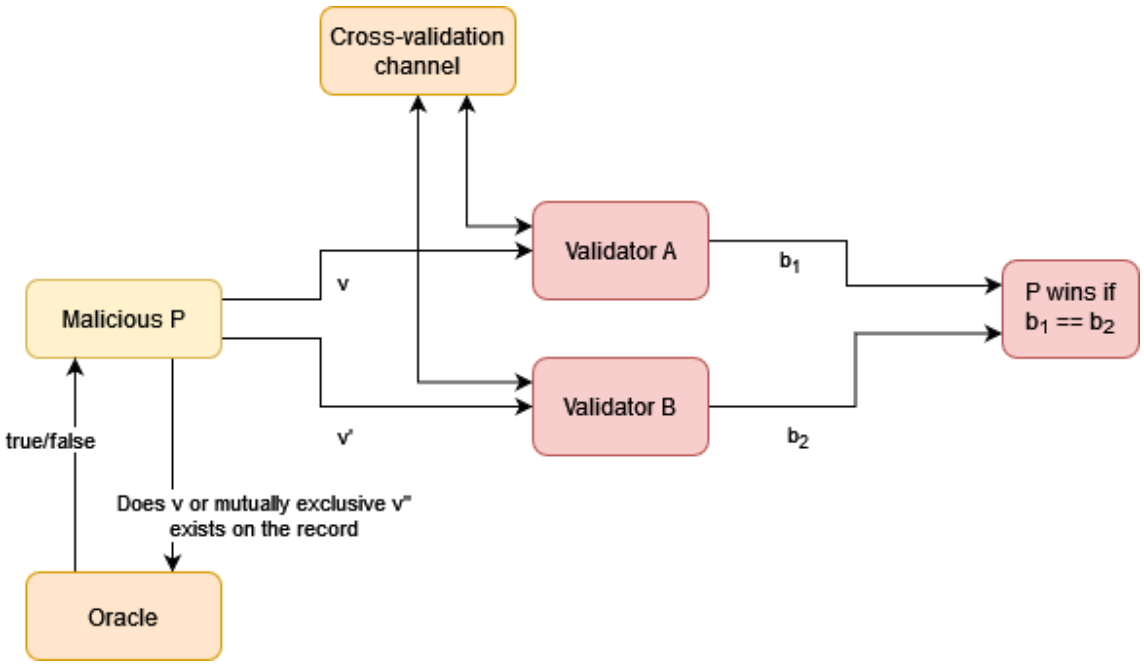


Fig. 4. Game Model for Data Integrity

The game comprises of two players:

1. A malicious *Proposer*: This adversarial player aims to append different instances of behavior to different vehicles. The *Proposer* can propose two mutually exclusive values v and v'. The *Proposer* can query the oracle to ensure that v and v' are not already blocked by some third mutually exclusive v" already on the blockchain.
2. Two *Validators*: Each player vehicle will evaluate the proposal and either accept (by outputting b = 1) or reject it (b = 0).

The game model design for $Integrity - Validation_{\pi,P}(n)$ is defined as below:

1. A *Proposer* proposes two different, mutually exclusive values v and v', to two different validators. Both values must be possible ground truths.
2. The *Proposer* has access to an oracle, which informs it whether a proposal already exists in the record V. However, the proposer cannot propose an existing v i.e v, v' $\notin$ V. It also can't propose v, v' that is blocked by a third mutually exclusive value v" in V.
3. Each *Validator* can communicate with other *Validators* through some Cross-validation channel
4. The *Validator i* outputs $b_i = 1$ if it accepts the proposal, or 0 if it rejects the proposal.
5. The *Proposer* wins the game if $b_1 = b_2$.

Here, we propose that the scheme is *Integrity-Validatable* if and only if for any probabilistic polynomial time adversarial *Proposer*, there exists a negligible function such that:

$$Pr[Integrity - Validation_{\pi,P}(n) = 1] \leq negl(n)$$

This game is simpler than the previous one since several assumptions hold.

1. Both values v and v' pass the multi-agent consensus game, meaning that given the same set of *Validators*, they will agree on appending v and v', if presented individually

2. Data integrity is only considered post-consensus. That is, we are only concerned about the integrity of the data if it can be added to the records. If no consensus can be reached about the new proposal, then data integrity is not of concern.
3. Following (2), we can assume all *Validators* are good, since malicious ones will always reject the proposals if they are not ground-truth, namely v and v'.

Model Analysis. Proving that our proposed system maintains Integrity-Validation is much more simpler in this case. Since we can already guarantee that we only need to handle good values, we only need to make sure one good value from 2 or more possible good values are added. Blockchain, as our cross-validation channel, easily guarantees that.

Assuming that after confirming that some value is good, each good agent attempts to add their value to the blockchain via mining. In the vast majority of cases, one of the two agents adds their value to the blockchain first (setting b = 1), and broadcasts that successful value. The other agent, upon seeing the successful value is mutually exclusive to their own value, will cease mining and accept this alternate value, while rejecting their own value (and setting b' = 0), preventing the case where $b_1 = b_2 = 1$. And since upon accepting a particular value as a possible ground truth, each *Validator* will mine until either they succeed or are beaten by another *Validator*, this prevents the case where $b_1 = b_2 = 0$.

Thus, b1 will always be different from b2.

In the unlikely case where both *Validator* successfully mine simultaneously, resulting in divergent transaction histories, they may work on their separate histories with the next set of values until one of them adds the next value first. Then all *Validator* switch to the longest history.

8 Conclusion

With the rapid development of autonomous vehicles, the security of the design becomes even more important. Autonomous vehicles are composed of various components and systems. Thus, there are a wide range of security issues that can be discussed.

In this project, we focused on aspects of security that are related to the decision making of a car. Hence, we discussed the achievement of consensus within autonomous vehicles and multi-agent integrity. We discussed consensus in two aspects, validation and unforgeability.

Validation is concerned with the unconditional acceptance of a valid proposal, and the unconditional rejection of a malicious one. Unforgeability is concerned with ensuring that an established consensus is immutable.

Multi-agent integrity is concerned with the recognition of the proposal closest to objective truth. Achieving the aforementioned consensus and selecting the most accurate information can help autonomous vehicles make more informed decisions.

Through the use of our game models, we demonstrated how these two security properties can be upheld. However, our proposed system relies on certain assumptions to ensure the security properties. Rather than serve as an exact blueprint for how these properties can be upheld, we hope that our models encourage further study into the implementation of systems to uphold them.

References

Autotalks. (2020, November 24). DSRC technology: 802.11p standard: Its-G5 and etsi its-G5, IEEE802.11P. Autotalks. https://www.auto-talks.com/technology/dsrc-technology/#:~:text=DSRC%20(Dedicated%20Short%2DRange%20Communications,involving%20cellular%20or%20other%20infrastructure.&text=The%20receiving%20vehicle%20validates%20the%20authenticity%20of%20the%20received%20messages.

Clark, M. (2021, September 9). Blockchain, explained. The Verge. https://www.theverge.com/22654785/blockchain-explained-cryptocurrency-what-is-stake-nft.

Google. (2021, November 1). Google Books Ngram Viewer – Google Product. Google Ngram Viewer. https://books.google.com/ngrams/graph?content=self-driving+car&year_start=2000&year_end=2019&corpus=26&smoothing=3&direct_url=t1%3B%2Cself%20-%20driving%20car%3B%2Cc0

Gringer, B. (2020, April 30). History of the autonomous car. TitleMax. https://www.titlemax.com/resources/history-of-the-autonomous-car/#:~:text=History%20of%20Autonomous%20Cars,made%20this%20concept%20a%20reality.

IBM. (n.d.). What is Blockchain Security? IBM. https://www.ibm.com/topics/blockchain-security.

Keshariya, A. (2020, May 9). Game theory in consensus protocols for Blockchain. LinkedIn. https://www.linkedin.com/pulse/game-theory-consensus-protocols-blockchain-astha-keshariya-ph-d-/.

Ondruš, J., Kolla, E., Vertaľ, P., & Šarić, Ž. (2020). How Do Autonomous Cars Work?. Transportation Research Procedia, 44, 226-233.

Synopsys. (n.d.). What is an autonomous car? – how self-driving cars work. Synopsys. https://www.synopsys.com/automotive/what-is-autonomous-car.html.

Contribution

Name	Work	Meeting Participation (absence only counted without informing in advance)
TSAI, HSIAO-HAN	Overall structure and formatting, Defense Mechanism, Assumptions, Game Model (Consensus), Analysis (Consensus), Conclusion	100%
Banerjee Aditya	Introduction, Definition of Proposed System, Game Model (Consensus)	100%
Tng Jun Wei	Abstract, Challenges, Game Model (Integrity), Presentation Video Editing	100%
Ho Wei Jun Sherman	Security Properties, Game Model (Integrity), Analysis (Integrity)	75%

The Future of Road Junctions in the Era of Self-Driving Cars

Chew Cheng Yap, Hwang Yong Kang, Ong Yi Jie Melvin, and Paw Su Yuan

National University of Singapore

Abstract. Self-driving cars have long been a vision for the automotive industry. With the rapid advances in Artificial Intelligence (AI) technology, it could be a possibility in the near future.

Waymo, Google's sibling company, has been developing and testing self-driving vehicles, and has covered over 20 million miles on public roads since being created in 2009 [1]. Its promising start could bring about revolutionary changes on our roads, such as lesser accidents, smoother roads, lower costs for consumers and less pollution.

When such self-driving options arrive, our roads infrastructure will have to be improved to accommodate these cars, as concerns over navigation and the circumvention of dynamic road situations will arise.

The self-driving cars will also have to be smart enough to navigate traffic conditions such as the use of shared junctions and intersections.

Our paper will discuss the various security properties to ensure a seamless traffic flow and suggest a scheme to safely maneuver around other vehicles in a fair manner. We will also explore the vulnerabilities of such a system and discuss its limitations.

Keywords: Fairness · Self-driving · Traffic

1 Introduction to the background/context

In the year 4236, the leading car manufacturing company TELSA has monopolized the self-driving car industry with every household owning at least 1 TELSA car. With their constant innovation and extensive use of artificial intelligence, the need for human drivers quickly became obsolete. Automotive technology has brought about cleaner air, thanks to reduced emissions, lesser congestion on the roads and reduced loss of human lives from accidents.

It took TELSA many years to perfect its system due to a variety of problems such as the numerous and dynamic road situations. Nonetheless, TELSA has come up with many sophisticated solutions to get to where we are today. One such solution was to ensure a fair way to schedule cars that arrive at a shared

road junction.

The objective of this system is to select which car should go first in a shared road juncture. When n cars arrive at a shared junction, each car will be notified that there are n cars waiting on the road, they will then send a response to an unbiased arbiter, part of TELSA's system to select which of the n cars shall go first. TELSA wanted to make sure that in the event of a malicious car, or perhaps a faulty system, the scheduler would still give a fair outcome and there would not be a widespread catastrophe. Intersections are much more efficient as traffic flows are seamlessly guided according to efficiency, instead of being designed with human behaviour in mind.

In this report, we shall explore an unbiased way to select which car should go first in a shared road juncture by providing an overview of our scheme. At the same time, defining certain security properties as well as some limitations of this system. Lastly, we give a possible construction of our scheme to ensure fairness in this scheduling regime.

2 Fair Juncture Scheme

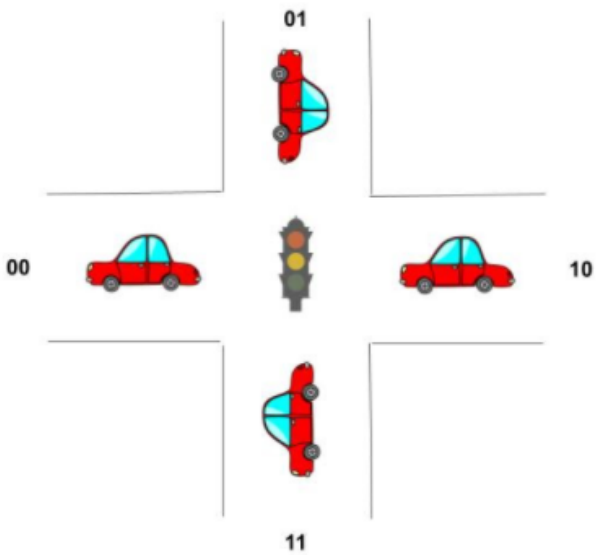


Fig. 1. 4 cars at a junction

In our system, we have a scheme to ensure a fair manner to choose which car will get to go first when arriving at a shared road junction. On arrival of n cars, the system will query the cars for a random key to be used in determining which car shall go first at the shared junction.

- $Car(n)$: On knowing the number of cars at a junction n, outputs a random key k where $k \in \{0, \ldots, n-1\}$ with size $|k| = log_2(n)$ bits
- $Output()$: On receiving n number of inputs keys k. Outputs a final value b which represents which is the car that will go first.

A Fair Juncture scheme (Car, Output) with n number of cars is fair if and only if for any n number of cars, $Pr[b = Car_n] = \frac{1}{n}$

In our project, for simplicity, we will only consider $n = 4$ cars when defining our security properties. In Figure 1, there are 4 cars labelled 00, 01, 10, 11 and the unbiased system will output a value within the space $\{00, 01, 10, 11\}$ to determine which car will go first in the shared junction.

3 Security Properties

3.1 Security property 1: Secure against 1 malicious car

Consider a scenario where 3 innocent cars will uniformly output a random 2-bit string, while there is 1 adversary car that can output whatever it wants. Note that the adversary car has no eavesdropping capability on the other 3 normal cars. We denote an innocent vehicle as a vehicle that adheres to the Fair Junction scheme defined earlier.

Game
We want to prove that adversaries cannot gain the system by always moving first. The $Pr[b = Car_n] = \frac{1}{4}$ for our 4 cars scenario.

Game Definition

1. Innocent cars 00, 01 and 10 generate a random 2-bit value k_0, k_1 and k_2.
2. Malicious car 11 is able to output a 2-bit value of its choice, k_3 and outputs a value b'
3. The output b is generated by doing an XOR operation on k_0, k_1, k_2 and k_3 where $b = (k_0 \oplus k_1 \oplus k_2 \oplus k_3)$
4. The output of the game is defined to be 1 if $b' = b$ and 0 otherwise. In the former case, the adversary succeeds in guessing which vehicle will proceed first.Thus, we write $Pr[BasicProtection_{A,\sqcap}^{OneAdvSecure} = 1] = \frac{1}{4}$

Definition

A Fair Juncture Scheme $\sqcap$ is fair in choosing a car if, for any adversary car with no restriction on its output and no eavesdropping capabilities, the adversary cannot succeed with more than $\frac{1}{4}$ probability.

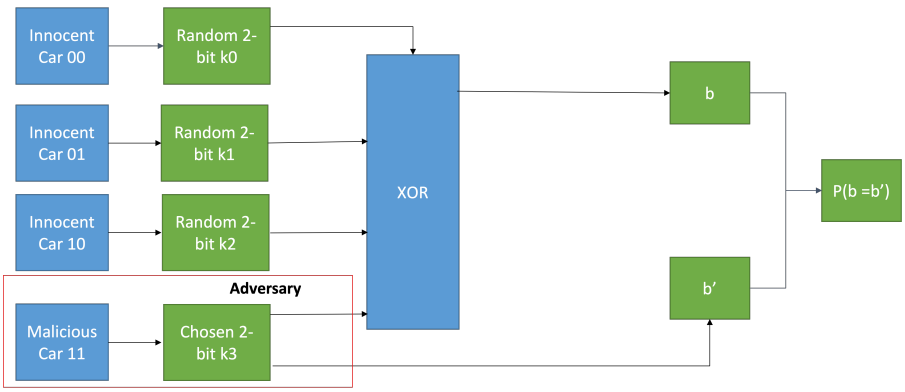


Fig. 2. Secure against 1 malicious car

3.2 Security property 2: Secure against collusion of multiple malicious cars

Consider the scenario where there is more than one adversary at the cross-junction. One desired security property would be that multiple adversary cars are unable to collude and tamper with the outcome of the system as long as there exists at least one "innocent" vehicle at the cross junction. Tampering of the game involves outputting a chosen 2-bit value and colluding with other adversary cars. We denote an "innocent" vehicle as a vehicle that adheres to the Fair Junction scheme defined earlier.

Game
We want to prove that multiple adversaries cannot exploit the system by moving first. The $Pr[b = Car_n] = \frac{1}{4}$ for our 4 cars scenario.

Adversarial Game

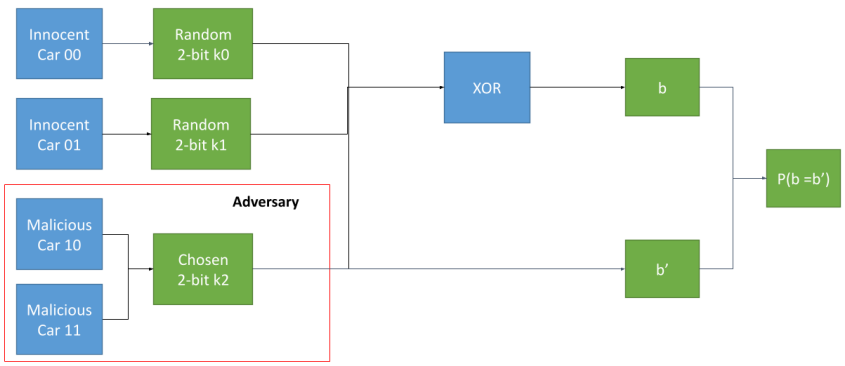


Fig. 3. Secure Against Multiple Malicious Cars

Game Definition

1. Malicious cars 10 and 11 are able to output a 2-bit value of their choice. They decide to collude, thus formulating a single 2-bit value of their choice, denoted by k_2 and outputs a value b'
2. Innocent Car 00 and 01 generate a random 2-bit value k_0 and k_1.
3. The output b is generated by doing an XOR operation on k_0, k_1 and k_2
4. The output of the game is defined to be 1 if b' = b and 0 otherwise. In the former case, the adversary succeeds in guessing which vehicle will proceed first.Thus, we write $Pr[AdvancedProtection_{A,\sqcap}^{MultipleAdvSecure} = 1] = \frac{1}{4}$

We can reduce this scheme to a one time pad construction using Karp Reduction. $k_0 \oplus k_1$ is equivalent to the key in the one-time pad. k_2 and output b correspond to the plaintext and ciphertext in the one-time pad respectively. This reduction can be done in polynomial time. Hence, if the one-time pad is perfectly secure, our scheme is perfectly secure.

Definition 2.3(a) states that for any two messages $m, m' \in$ M the distribution of the ciphertext when m is encrypted should be identical to the distribution of the ciphertext when m is encrypted [2].

Equivalently, in our scheme, the probability of the outcome when k_2 is forged or when collusion happens is equal to the probability of the outcome when k_2 is

generated randomly which is equal to $\frac{1}{4}$

Definition

A Fair Juncture Scheme $\sqcap$ is fair in choosing a car if, for multiple adversary cars (up to n-1) with no restriction on its output and allowed to collude, the adversary cannot succeed with more than $\frac{1}{4}$ probability.

$$Pr[AdvancedProtection_{A,\sqcap}^{MultipleAdvSecure} = 1] = \frac{1}{4}$$

3.3 Security property 3: Protection against faulty/malicious systems

Consider a scenario where the system is faulty and has the ability to **override the input bits of any 1 car during an operation**. Note that the system has **no eavesdropping capabilities** and does not know the input bits presented to it before it overrides 1 of the given inputs.

We want to protect against such a malicious adversary such that the adversary is not able to successfully override the output bits, i.e. the $Pr[b = Car_n] = \frac{1}{4}$ for our 4 cars scenario.

Adversarial Game

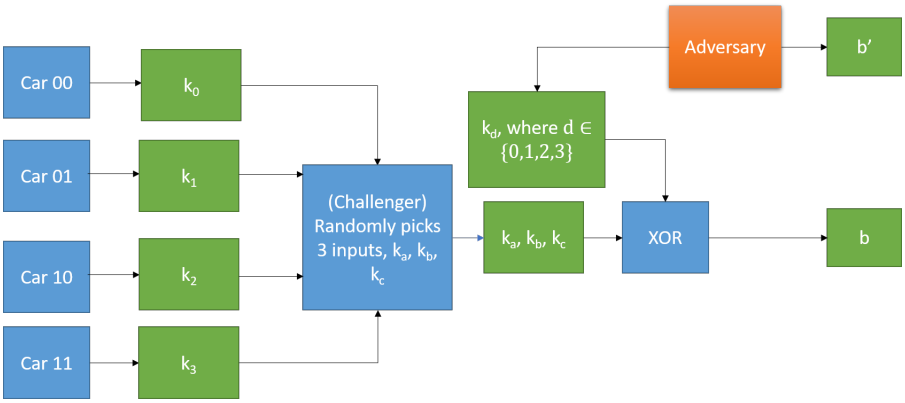


Fig. 4. Protection Against Faulty System

1. All 4 cars are fair and outputs a random 2-bit value (k_0, k_1, k_2, k_3)
2. Challenger randomly picks 3 values (k_a, k_b, k_c) from step 1
3. Adversary outputs a 2-bit value k_d and passes it to the challenger
4. Challenger XORs the 4 inputs and gives an output b
5. Adversary outputs a 2-bit value, b'
6. Adversary succeeds if b = b', we write $Pr[FairPlay_{A,\sqcap}^{FaultySystem} = 1]$

Definition
A Fair Juncture Scheme $\sqcap$ is fair in choosing a car if for any malicious systems(A), the adversary cannot succeed with more than $\frac{1}{4}$ probability.

$Pr[FairPlay_{A,\sqcap}^{FaultySystem} = 1] = \frac{1}{4}$

4 Construction for our secure scheme

Let n be the number of cars arriving at a shared road junction. Let l be the number of bits to represent the number of cars at the road junction such that $l = log_2(n)$. The keyspace K and output space B are equal to $\{0,1\}^l$.

$Car(n)$: Upon arrival of n cars at a junction, each car chooses a l bit key k from key space $K = 0, 1^l$ according to the uniform distribution (i.e, each of the strings in the space $\{0,1\}^l$ is chosen as the key with probability exactly $\frac{1}{n}$).

$Output()$: Given n number of inputs $(k_0, ..., k_n)$, the system XORs all the inputs together and outputs a bit $b := k_0 \oplus ... \oplus k_n$ where b is in the output space B.

Proof. For any Car_n,
$Pr[Car_n$ chooses any $k] = \frac{1}{n}$, where k $\in$ key space K since k is chosen uniformly at random.

$Pr[k_0 \oplus ... \oplus k_n = Car_n]$
$= Pr[b = Car_n]$
$= \frac{1}{n}$ (since $b \in B, KandB \in \{0, \ldots, n-1\}$)

5 Failure of the scheme: Vulnerability to eavesdroppers

So far, we have seen how our scheme is secure against an adversary or multiple adversaries that try to tweak the game in their favour. Our scheme is also secure against potential faults that overrides any one car's value.However, our scheme is not secure against an adversary that has eavesdropping capabilities and is able to know the values of the other 3 cars beforehand. An eavesdropping adversary is able to produce a value such that our scheme will output whatever the adversary wants with absolute certainty.

We want to show that if the adversary has eavesdropping capabilities and is able to read the output of the 3 other cars, then, given that the adversary can output whatever it wants, the scheme fails (i.e. probability = 1 != $\frac{1}{4}$).

To prove this, let us construct an adversary with the following properties:

Construction of Adversary

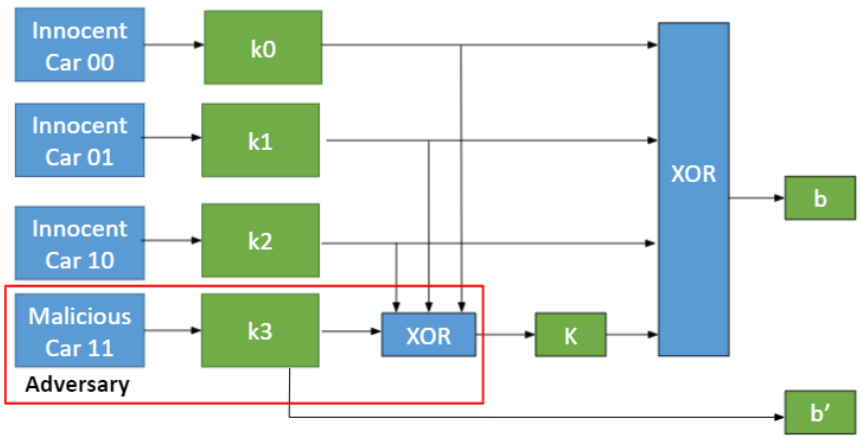


Fig. 5. Failure of the Scheme

1. Cars 00, 01, and 10 output a random 2-bit value k0, k1, and k2 respectively
2. Let $b' = k_3$ be the 2-bit value Malicious Car 11 wants the Scheme to output
3. Malicious Car 11 computes $K = (k_0 \oplus k_1 \oplus k_2 \oplus k_3)$ and send K to the Scheme
4. The Scheme will compute $b = (k_0 \oplus k_1 \oplus k_2 \oplus K) = ((k_0 \oplus k_0) \oplus k_1 \oplus k_1) \oplus (k_2 \oplus k_2) \oplus k_3) = k_3 = b'$
5. Hence, the Malicious Car 11 is able to get the Scheme to output whatever values it wants with a probability of 1 (i.e. $b == b'$)

As shown, there exists an adversary with eavesdropping capabilities can force the system to output whatever value it wants. Thus, our scheme is not secure against eavesdroppers.

6 Summary/Conclusion

To summarise, a Fair Juncture scheme (Car, Output) with n number of cars is fair if and only if for any n number of cars, $Pr[b = Car_n] = \frac{1}{n}$. Additionally, it should possess certain security properties such as (1) secure against 1 malicious car, (2) secure against collusion of multiple malicious cars and (3) offers protection against faulty/malicious systems.

In the distant future where self-driving vehicles will dominate the roads, a fair juncture scheme will ensure that vehicles will not suffer from starvation where

they wait aimlessly for long periods, especially at a busy cross junction.

However, the scheme discussed in this paper seeks only to provide a simplistic view of how a traffic cross-junction works. It assumes that only vehicles from a certain direction can be travelling concurrently and fails to take into account instances where vehicles from different directions can travel simultaneously. Furthermore, as discussed in the limitations above, adversaries with the ability to eavesdrop can easily manipulate the scheme to output whatever value it wants. For our scheme to work and be truly fair, there must not be any way for a malicious car to know beforehand what values the other cars are going to use.

References

[1] Reuters. (2020). Waymo self-driving vehicles cover 20 million miles on public roads. Reuters. https://www.reuters.com/article/us-autonomous-waymo-idUSKBN1Z61RX.

[2] Katz, J., Lindell, Y. (2015). Introduction to modern cryptography (Second edition). CRC Press/Taylor Francis.

The security Analysis of the Privacy Goals of BlueTrace Protocol*

Dianne Loh Wen Hui, Li Yiwen, Oen Qi Han, Kenson, and Woo Jian Zhe

National University of Singapore

Abstract. There is growing interest in technology-enabled contact tracing, the process of identifying potentially infected COVID-19 patients by notifying all recent contacts of an infected person. In this project, we focus on the security Analysis of the Privacy Goals of BlueTrace Protocol. BlueTrace protocol uses a back-end service to issue each user with the tracing application a unique and randomised UserID that can be associated with the user's contact number. In order to achieve untraceability, protection of infected users and protection of interaction information that related to privacy goals of BlueTrace protocol, two games are formulated, constructed and proofed. Our first game model formulates trace unlinkability and proofs that An unauthorised entity with access to the data broadcast from a user should not be able link two encrypted anonymous token to the same person. Our second game model formulates Chosen TempID attack. Both our game models are closely related to real life.

Keywords: Privacy · Security · BlueTrace.

1 Introduction

Due to the highly contagious nature of the virus, social distancing is one fundamental measure which has already adopted by many countries. At the technical level, this prioritises contact tracing solutions, which can alert the users who have been in close contact with the infected persons and meanwhile allow heath authorities to take proper actions. Contact tracing is the process of identifying, assessing, and managing people who have been exposed to someone who has been infected with the COVID-19 virus. As contact tracing technologies analyze sensitive information about citizens' locations, they raise civil liberties concerns. In this paper, in order to achieve untraceability, protection of infected users and protection of interaction information that related to privacy goals of BlueTrace protocol, two games are formulated, constructed and proofed.

1.1 Related Works

At the height of the pandemic last year, various methods of contact tracing were hotly debated as governments around the world looked for a means to control

* Supported by National University of Singapore.

the infection rates. Consequently, several publications on the security analysis of contact tracing methods have been released for the governments' considerations. Mentioned below are some of the related works that our team has examined and taken inspiration from at the time of writing this report.

Several security and privacy concerns about contact tracing were discussed in Centralized or Decentralized? The Contact Tracing Dilemma by Vaudenay[1], as he weighed the benefits and ramifications for both centralized and decentralized contact tracing. In the report, he also mentioned multiple attacks that could be conducted on each system. One such example is the False Encounter: The Lazy Student Attack: A replay attack on the centralized system protocol that raises an at-risk alert to the victim, inducing substantial stress upon the victim. He eventually concluded that both systems suffered from privacy issues that cannot be rectified and called for solutions that involved a hybrid of both systems.

Danz-Derwisch-Lehmann-Puenter-Stolle-Ziemann[2] gave a detailed security analysis of decentralized contact tracing protocols in Provable Security Analysis of Decentralized Cryptographic Contact Tracing. In the analysis, they introduced the concepts of Pseudonym Unlinkability and Trace Unlinkability in the context of contact tracing. They proceeded to formulate a security game to prove these properties about decentralized contact tracing models. In contrast, our work aims to prove similar security and privacy goals in hybrid contact tracing.

Cho-Ippolito-Yu[3] discussed the privacy implications of the TraceTogether centralized contact tracing method employed by the Singaporean government. They suggested potential measures to mitigate the privacy concerns without sacrificing its effectiveness. This gave us valuable insights on the problems that are specific to the BlueTrace protocol.

2 Hybrid Contact Tracing Scheme

Overview of Hybrid Contact Tracing Scheme In the Hybrid Contact Tracing (HCT) Scheme, the data collection follows a decentralised approach as individuals exchange tokens without the involvement of the Health Authority (HA), while the identification of close contacts after diagnosis follows a centralised approach, where infected individuals reveal their entire contact history to the Health Authority [5].

In the HCT Scheme, a User U has an application App_U installed in their smart phone. The App_U is registered with the HA and is given a unique identifier UID. The application App_U is regularly loaded with a list L_{out}^U of temporary identifiers $t_1 ... t_n$ to be released in a specified order. During an epoch, App_U repeatedly broadcasts its temporary identifier t_i^U to nearby devices. At the same time, App_U receives temporary identifiers t_j^V from App_V of nearby User V and

keeps track of the received temporary identifiers in the list L_{in}^{U} [1].

When User U is diagnosed with COVID, User U retrieves a token from the HA, which allows him to upload his contact history L_{in}^{U} to the HA. From the temporary identifier in the contact history t_j^V, the HA can query its database to associate it with the User V and hence identify the close contact User V.

Formal Definition of the Hybrid Contact Tracing Scheme HCT Scheme consists of the protocols $(Register, GenTempID, Exchange, Report, Verify)$.

1. $Register$: An application App_U of User U registers with the HA. The HA sets a UID_U for App_U. Depending on the construction, the UID_U may be associated with a personally identifiable information that links to U, such as a phone number, or just a random pseudonym.
2. $GenTempID(UID_U, e) = T_e^U = t_i ... t_n$: Generates a list of temporary identifiers for the User U and are only valid for the epoch e. We use e to represent some time interval or timing information, but the actual representation depends on the implementation.

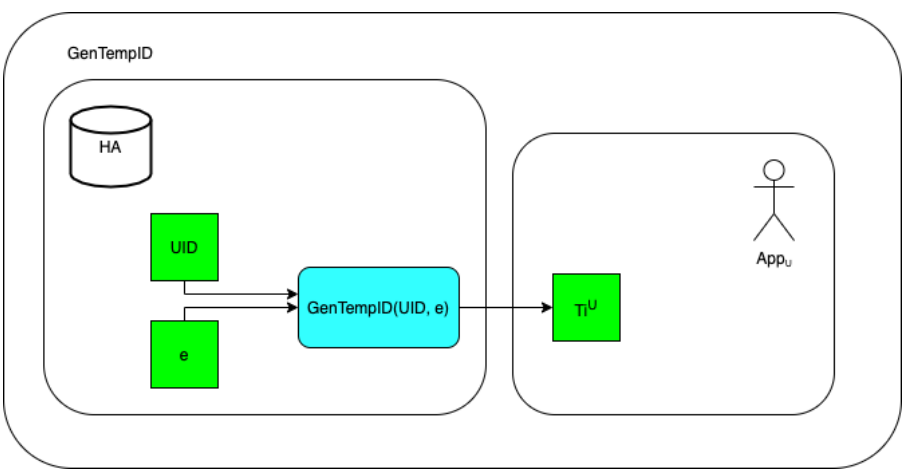


Fig. 1. GenTempID

3. $Exchange$: During the epoch e, the application App_U continuously broadcasts $t_i^U \in T_e^U$ to nearby devices. At the same time, it receives $t_j^V \in T_e^V$ from nearby application App_V from User V and stores it in L_{in}^{U} together with some coarse time information.
4. $Report$: When User U is diagnosed, he uses an authorisation token from the HA to upload its contact history L_{in}^{U}. From its database, the HA will retrieve the UID associated with each temporary identifier t_i^V in L_{in}^{U} (i.e. reverse of $GenTempID$) and hence identify close contact User V.

5. $Verify$: Verify that the temporary identifier is valid and unmodified within its time interval. Outputs 1 if valid and 0 if invalid. If the temporary identifier is invalid, do not treat User V as a close contact.

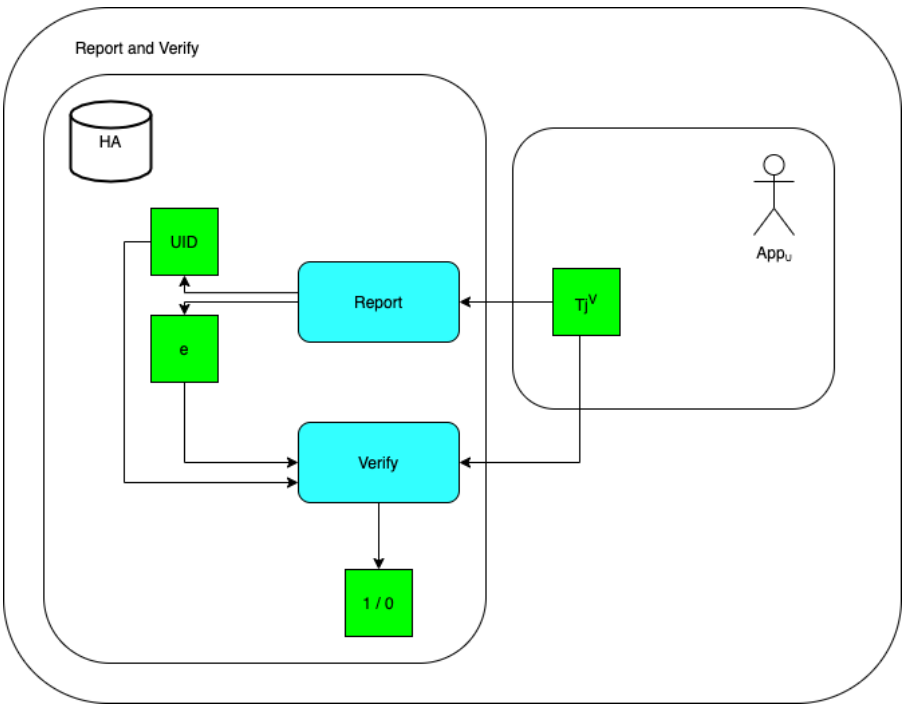


Fig. 2. Report and Verify

3 Security Properties

Aspects of both centralised and decentralised contact tracing schemes are important in determining the relevant security properties for a hybrid contact tracing scheme. We will be focusing on two specific security properties relating to privacy, Trace Unlinkability and security against Existential Forgery under a Chosen Token Attack. We have come up with formal definitions for these two formulations.

3.1 Trace Unlinkability

Users are linked to their pseudonym which will only be valid for a certain time interval. An unauthorised entity with access to any data broadcast from a user

should not be able to link two encrypted anonymous userIds. Alternatively, given two pseudonyms from different epochs, the attacker should not be able to determine if they originated from the same user. As such, the attacker should not be able to track an individual user based on his data across any significant period of time. Trace unlinkability keeps the privacy of infected users secure during tracing.

We present another equivalent experiment-based definition of trace unlinkability. The experiment involves an adversary obtaining a token from a user, and subsequently observing more tokens and trying to guess which of the tokens belong to the same user. This is identical to what a real attack would resemble as an adversary could trace a victim's location by obtaining tokens from various locations and trying to match it with a valid token obtained from the victim previously.

In this context, we consider the experiment: An adversary A obtains a valid token t_u at *epoch* e_i from the victim, user U. A random token generator uniformly picks $b \in \{0, 1\}$ at random and outputs t_0 or t_1 to the adversary. t_0 represents a valid token not from user U whereas t_1 is another valid token of user U at a different *epoch* e_j. Finally, A outputs a "guess" as to whether the token from the random source belongs to user U or not. A succeeds if it manages to guess correctly. We say that a contact tracing scheme is trace unlinkable if and only if no probabilistic polynomial-time adversary A can succeed with probability better than $\frac{1}{2} + negl(n)$.

Formally, we let $\pi = (\text{GenTempId})$, adversary A is just a poly-time algorithm, and any value n as the security parameter.

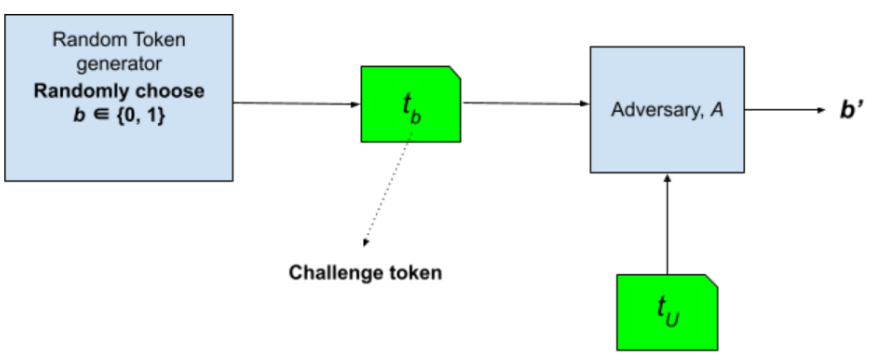


Fig. 3. Trace unlinkability game

The Trace Unlinkability Experiment Trace-unlink$_{A,}\pi(n)$:

1. Adversary A obtains token t_u = GenTempId(UID_u, n, e_i) from user U.
2. A uniform bit $b \in \{0, 1\}$ is chosen by a random token generator. Token t_0 = GenTempId(UID_v, n, e_j) or t_1 = GenTempId(UID_u, n, e_j) is generated and passed to A. We refer to this output as the challenge token.
3. A outputs bit b'.
4. The output of the experiment is defined to be 1 if b' = b and 0 otherwise. We write Trace-unlink$_{A,}\pi(n) = 1$ if A succeeds.

Definition 1. *A contact tracing scheme = (GenTempId) is trace unlinkable if and only if for all probabilistic polynomial-time adversaries A, there is a negligible function negl such that, for all n*

$$\Pr[\text{Trace-unlink}_{A,}\pi(n) = 1] \leq \tfrac{1}{2} + negl(n)$$

3.2 Existential Forgery Under a Chosen Token

An attacker may have access to valid Tokens (TempID) from a real encounter where data has been transmitted or exchanged as per the contact tracing protocol. Given these valid tokens, given that the adversary does not have the decryption key for the token, he should not be able to forge a new original token that would be construed as valid by the system. A forged valid token with a different ID could enable an impersonation attack, while one with a different start and end time would enable a replay attack.

We present an experiment for this formulation, with 2 varying levels of security requirements. The first security requirement is more stringent/stronger - can the adversary forge any valid TempID for any current or future epoch. We name this existential forgery under a chosen Token.

For this, we consider the experiment: An adversary A has an oracle O that will generate a valid token t_i (TempID) of any query e_i, which represents the start timing of any epoch. A succeeds if it manages to forge any valid TempID t_f for any epoch that is valid for current or future epochs. We call this existential forgery for any chosen TempID.

We notice that the adversary is able to reuse the same tokens as created from the oracle, as long as the validity of the token has not expired yet, i.e. the current time is still within the epoch of the created token. This motivates an additional second winning condition for the adversary where the set of created TempIDs based on the specific queries to the oracle will be saved. A can now only succeed if t_f is not part of the set of queries and corresponding tokens generated by the oracle within the experiment.

The first game: Token-forge$_{A,\pi}(n)$

1. Generate key = $Gen(1^n)$
2. Adversary has oracle access. Adversary A obtains token t_i = GenTempId(UID_a, n, e_i) from Oracle O with input e_i.
3. Adversary outputs a forged Token t_f
4. The output of the experiment is defined to be 1 if t_f is a valid token for the current epoch, i.e. t_f = GenTempId(UID_f, n, e_f) for some f and 0 otherwise. We write Token-forge$_{A,\pi}(n) = 1$ if A succeeds.

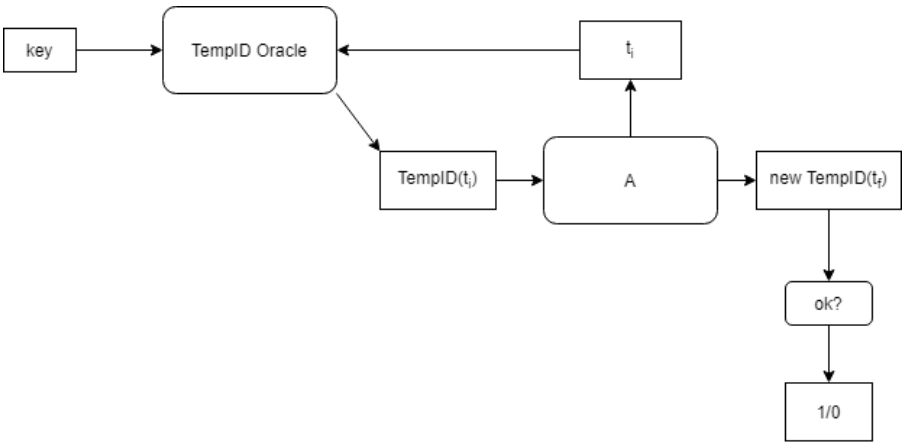


Fig. 4. Existential Forgery:Token-forge$_{A,\pi}(n)$

The second game: Token-sforge$_{A,\pi}(n)$:

1. Generate key = $Gen(1^n)$
2. Adversary has oracle access. Adversary A obtains token t_i = GenTempId(UID_a, n, e_i) from Oracle O with input e_i. Let Q be the set of all tokens generated by the oracle in this experiment.
3. Adversary will create a forged Token t_f
4. The output of the experiment is defined to be 1 if t_f is a valid token, i.e. t_f = GenTempId(UID_f, n, e_f) for some f AND $t_f \notin Q$ where Q is the set of TempIDs created from the Oracle and 0 otherwise. We write Token-sforge$_{A,\pi}(n) = 1$ if A succeeds.

Definition 2. *A contact tracing scheme Π = (GenTempId) is existentially unforgeable under an adaptive chosen-token attack, or simply secure, iff for any PPT A, there is a negligible function negl, such that for all n*

$$\Pr[\text{Token-sforge}_{A,\pi}(n) = 1] \leq \tfrac{1}{2} + negl(n)$$

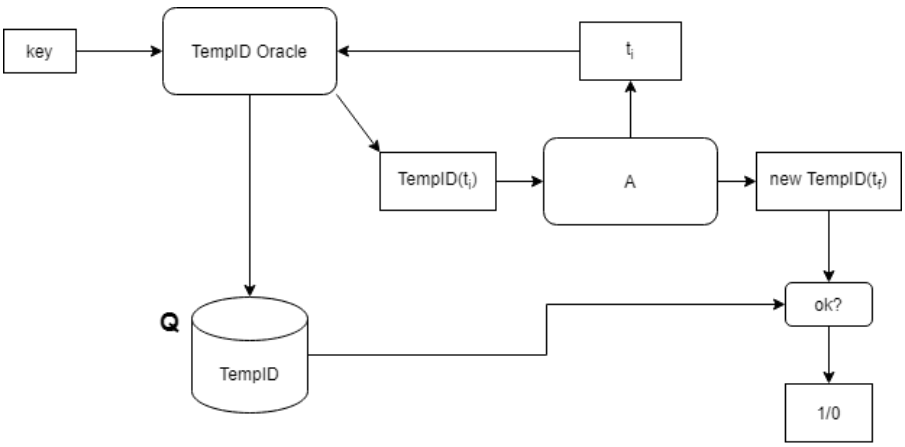


Fig. 5. Existential Forgery:Token-sforge$_{A,}\pi(n)$

4 Construction: BlueTrace

BlueTrace [6] is one construction of HCT scheme.

1. *Register*: Upon registration, the HA backend will associate a random unique identifier UID to the user's phone number. The phone number is the only personally identifiable information from the user.
2. *GenTempID*: For each temporary identifier t_i^U, the UID, created time and expiry time is encrypted symmetrically with authenticated encryption scheme AES-256-GCM using the HA secret key K and with a random Initialisation Vector (IV). Only the HA knows the secret key. The ciphertext is appended with the IV and an Auth Tag for integrity checks and then encoded with Base64. The final result forms the temporary identifier. The temporary identifiers have a lifespan of 15 minutes. This will be important in the discussion of the Chosen TempID Attack.
3. *Exchange*: Two nearby applications will exchange their temporary identifiers with each other via Bluetooth Low Energy. For example, App_U of User U will receive User V's temporary identifier t_j^V and store it in L_{in}^U as part of its contact history.
4. $Report, Verify$: When User U is diagnosed, he will upload his contact history L_{in}^U to the HA. The HA decrypts the TempID to retrieve UID_V and the timing information. The HA verifies that the authentication tag is valid and that the encounter timestamp for each TempID is within its valid time interval. It then identifies close contacts that were encountered within the sick period (e.g. 14 days) by mapping the TempID to the UID from its database.

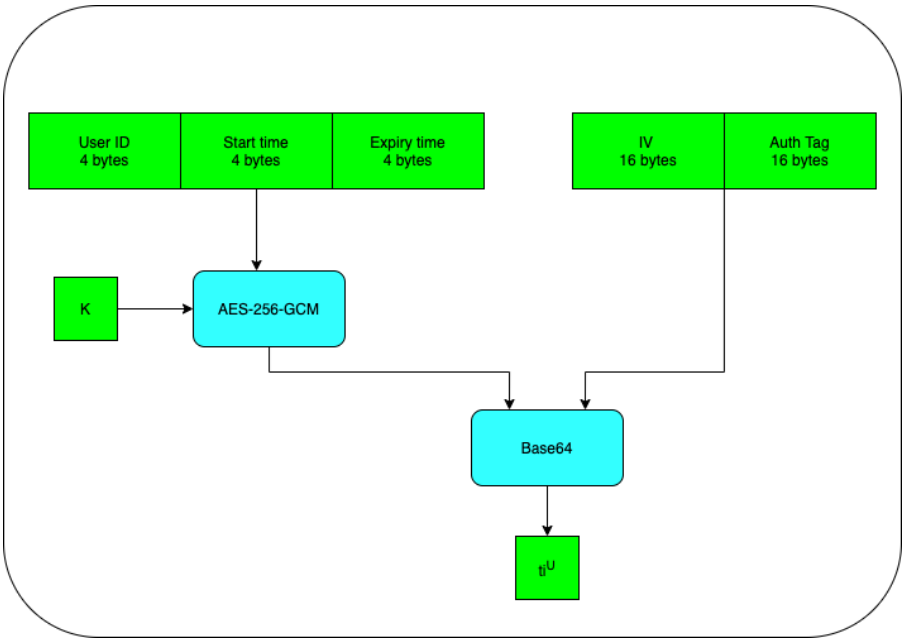


Fig. 6. BlueTrace GenTempID

5 Security Analysis of BlueTrace

This section analyses the security of BlueTrace with respect to Trace Unlinkability and security against Chosen TempID Attack.

5.1 BlueTrace: Trace Unlinkability

In this section, we aim to show that the BlueTrace protocol satisfies the desirable property of trace unlinkability. We do this by using the game-based definitions defined in section 3.

Theorem 1. *A contact tracing scheme π = (GenTempId) is trace unlinkable if the underlying encryption scheme in GenTempId is EAV-secure.*

Proof. We can show this through a reduction (see Fig. 7). Consider the following construction of the trace unlinkability experiment where π = (GenTempId) and GenTempId is the token generation algorithm $Enc_k(UID, e_i)$ and key k is randomly generated using a key generator Gen(1^n). The adversary A outputs a pair of (UID, e_j) for the token generator, one of which is for a user U and the other is not. The adversary also has another valid token t_u of user U which was generated at another time period.

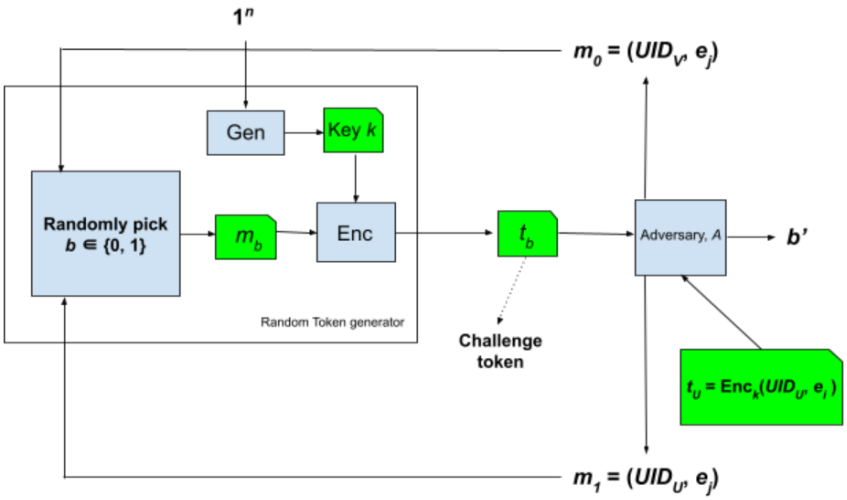


Fig. 7. π

We assume that π is not trace unlinkable but Enc_k is an EAV-secure scheme. This means $\Pr[\text{Trace-unlink}_{A,}\pi(n) = 1] > \frac{1}{2} + negl(n)$.

Since Enc_k is EAV-secure, by definition, this implies that the probability that adversary A is able to tell the difference between challenge token t_b and a previously obtained token t_u is $\leq \frac{1}{2} + negl(n)$. Consequently, Adversary A is able to win the trace unlinkability experiment π with probability $\leq \frac{1}{2} + negl(n)$.

This also means $\Pr[\text{Trace-unlink}_{A,}\pi(n) = 1] \leq \frac{1}{2} + negl(n)$, contradicting our initial assumption that π is not trace unlinkable. Thus, π must be trace unlinkable to begin with.

Corollary 1. *BlueTrace is trace unlinkable since the underlying encryption scheme is AES-256-GCM which is possibly EAV-secure.*

Galois/Counter Mode (GCM) has been proven by to be secure in the concrete model[4] when used with a block cipher that is indistinguishable from a random permutation. Currently, there are no concrete proofs to show that AES-256 is pseudorandom, but many people believe it to be pseudorandom. Additionally, there are also no known practical attacks on the AES scheme[7], which is one of the reasons why it is so widely used. Hence, it is safe to say that the BlueTrace is trace-unlinkable until AES is otherwise proven to be not pseudorandom.

5.2 BlueTrace: Security Against Existential Forgery under a Chosen Token Attack

In this section, we show that BlueTrace is not secure under Token-Forge but is secure under Token-sForge using the game-based formulations defined in section 3.

The first game: $\boldsymbol{Token - Forge_A\pi(n)}$ BlueTrace fails under this stricter security property because an adversary can win this game with probability 1.

Proof. Recall that in this game, the adversary is allowed to output a TempID that it has queried from the oracle before. We construct an adversary A as follows:

1. Query the oracle O by sending the start time e and receive a valid tempID t.
2. Output t.

Since t is a valid tempID as it is produced by the oracle, $Pr[Token{-}forge_A\pi(n)] = 1 > \frac{1}{2} + negl$. Hence, BlueTrace is not secure against the strong version of Existential Forgery Under a Chosen Token.

The second game: $\boldsymbol{Token - sForge_{A,\Pi}(n)}$ In this game, the adversary is not allowed to output a tempID $t \in Q$, where Q is the set of queries that it has received from the oracle previously. In the BlueTrace scheme, $Verify$ will verify the integrity of the tempID from the auth tag generated by AES-256-GCM and that it is valid if the created time is before the expiry date. To output another tempID, the adversary has to modify either the userID or the timing information and output a valid ciphertext with a valid authentication tag.

Theorem 2. *A contact tracing scheme $\Pi = (GenTempID, Verify)$ is secure under $Token - sForge_A\pi(n)$ if the underlying authenticated encryption scheme X in $GenTempID$ is Enc-Forge-secure.*

Proof. Suppose that the scheme is not secure under Token-sForge, that is, there exists an adversary A that can construct a valid TempID $t' \notin Q$ such that t' is valid (can be decrypted into a valid $< UID, t_i >$ and its authentication tag is valid). We will construct another adversary A' that will win the $Enc - forge_{A,X}(n)$ game. The adversary A' is constructed as follows:

1. Adversary A plays the Token-sForge game and outputs a valid token t' with probability $> \frac{1}{2} + negl$.
2. Output t' as the new ciphertext.

A' wins if A wins and vice versa because if t' is valid under Token-sForge, it is also valid under Enc-forge as it will decrypt to a valid plaintext and its authentication tag is valid. Hence, $Pr[Enc - forge_{A,X}(n)] = Pr[Token - sForge_{A,\Pi}(n)] > \frac{1}{2} + negl$.
However, this is a contradiction because scheme X is Enc-forge secure. Therefore, the HCT scheme must be Token-sforge secure.

Since the underlying authenticated encryption scheme that BlueTrace uses, AES-256-GCM, is Enc-forge secure [4], then BlueTrace is Token-sForge secure.

Relation to Real Life The attack model is realistic as the adversary has access to the TempIDs being transmitted over Bluetooth (the oracle). Since an adversary can win the first game with probability 1, an adversary can successfully carry out a replay attack by forwarding a valid token to other devices within the same epoch without modification. However, since it is secure under Token-sforge, an adversary is unable to modify the tempID or timing information. Hence, the effects of a successful attack is minimised as the adversary can only replay the message within the lifetime of a token, which is 15 minutes.

6 Conclusion

Contact-tracing will continue to be an important tool for public health officials to prevent the spread of COVID-19. To support this effort, governments and companies have been building technology solutions to be used at scale. We applied three privacy goals. Based on the research and security analysis of BlueTrace Protocol, in order to achieve achieve a series of security and privacy goals, we have formulated, constructed and proofed two game models. Our game models are closely related to real life.

References

1. Serge Vaudeny. Centralized or Decentralized? The Contact Tracing Dilemma. Cryptology ePrint Archive: Report 2020/531. IACR. http://eprint.iacr.org/2020/531
2. Noel Danz, Oliver Derwisch, Anja Lehmann, Wenzel Puenter, Marvin Stolle, Joshua Ziemann. Provable Security Analysis of Decentralized Cryptographic Contact Tracing. Cryptology ePrint Archive: Report 2020/1309. IACR. https://eprint.iacr.org/2020/1309.pdf
3. Hyunghoon Cho, Daphne Ippolito, Yun William Yu. Contact tracing mobile apps for covid-19: Privacy considerations and related trade-offs. Preprint arXiv:2003.11511 [cs.CR], 2020. https://arxiv.org/abs/2003.11511
4. McGrew, David A., Viega, John. The Security and Performance of the Galois/counter mode (GCM) of Operation. Proceedings of INDOCRYPT 2004. Lecture Notes in Computer Science. 3348. Springer. CiteSeerX 10.1.1.1.4591. doi:10.1007/978-3-540-30556-9_27. ISBN 978-3-540-30556-9.
5. Tedeschi, P., Bakiras, S., Pietro, R.D. (2020). SpreadMeNot: A Provably Secure and Privacy-Preserving Contact Tracing Protocol. ArXiv, abs/2011.07306.
6. Bay, J., Kek, J., Tan, A., Hau, C.S. (2020). BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders.
7. A. Biryukov and J. Großschädl. Cryptanalysis of the Full AES Using GPU-Like Special-Purpose Hardware. Fundam. Informaticae, vol. 114, no. 3–4, pp. 221–237, Aug. 2012.

A System for Securing NFTs' Originality and Ensuring their Availability

Huynh Thai Duong e0407657, Lim Jia Rui Ryan e0406796,
Ling Chen e0575752, Teng Dao Xiong e0404039

Abstract This paper attempts to formally define what a NFT system is, the desired security properties that a NFT system should possess as well as formal proofs on how a NFT system achieves such security properties, a formal construction of the NFT system based on the security properties and lastly we present limitations on the NFT system which are difficult to solve through cryptographic means.

INTRODUCTION

Non-Fungible Token (NFT) is a class of cryptocurrency that is based on smart contracts such as Ethereum [1]. NFT differs from classical cryptocurrencies [2] such as Bitcoin in its features and implementation. Bitcoin is a cryptographic token in which all the tokens are equivalent and indistinguishable. In contrast, NFTs are unique and cannot be exchanged with another NFT hence the term non-fungible. This makes NFTs suited for uniquely identifying any objects that could be represented digitally, to further elaborate, using NFTs based on Ethereum's smart contracts, a creator can easily prove the existence and ownership of digital assets such as videos, images, arts, etc. This uniqueness property of NFTs allows creators to sell and/or trade NFTs on any NFT market (such as MakersPlace, Rarible and OpenSea) and creators can earn royalties from such transactions.

In recent years, NFTs have gained much attention from both industrial and scientific communities. At the time of writing (November, 2021) the NFTs-related market has significantly increased compared to the start of the year (January 2021). The total number of sales reached 125,043 and their total amounts spent on sales reached 317,677,968.27 with primary-market sales reaching 63,632 and secondary-market (user-to-user) sales reaching 61,411. Besides the market data of NFTs, we are also seeing a growing ecosystem of NFTs. CryptoPunks is one of the first NFTs on Ethereum with 10,000 unique collectible characters called Punks, CryptoPunks also paved further developments of the ERC-21 standard, a token standard which supports much of NFTs systems today [5]. Beyond art and collectibles, games are also a beginning to look into leveraging on the NFT systems, CryptoKitties is one of the world's first games to be built on such systems [6]. There is no doubt that there is a significant interest surrounding NFTs where much of its products can be sold at extremely high prices, with the most expensive one being sold for US$69 million in March 2021 [6].

Despite the potential of NFTs having a significant impact on decentralised markets and the potential business opportunities in the exchange of NFTs, much of the NFT technologies are still in the infant stages. There are several potential challenges that need to be discovered and tackled. While there is a growing number of literature surrounding NFTs, formal definitions on what an NFT system is as well as formal studies on the security aspects of NFTs are minimal.

FORMAL DEFINITION of NFT SCHEME

A Non-Fungible token scheme consists of 5 probabilistic polynomial time algorithms (**GEN**, **MINT**, **STORE**, **TRANSFER**, **REVOKE**) & 3 entities a **Blockchain** β, **User** U and **Storage** DB_{ext} such that:

1) The key generation **GEN** takes as input a security parameter 1^n and outputs a pair of public and private keys$< P_k, S_k >$ for the NFT creator
2) The minting algorithm **MINT**, takes in as input a digital asset data D, a digital signature σ and public key P_k. It outputs a transaction t to the blockchain for recording, and to the user
3) The store algorithm **STORE**, takes in as input data of arbitrary length $D \in \{0,1\}^*$ and a database DB_{ext} and outputs a URL l that a user u can use repeatedly to access the digital asset
4) The trade algorithm **TRADE**, takes in as input a token Id Idx and a secret key S_k and outputs a transaction t to the blockchain for recording, and to the user
5) The revoke algorithm **REVOKE,** takes in as input a token Id Idx and a secret key S_k and outputs a transaction t to the blockchain for recording, and to the user

SECURITY PROPERTIES AND EXPERIMENTS

1. Non-Counterfeitability

The NFT scheme should be non-counterfeitable in the sense that nobody should be allowed to mint an NFT in somebody else's name without their authorisation. For example, an adversary might be motivated to mint an NFT of a random digital art and claim that the digital asset is created by Beeple, the artist who sold his artwork's NFT for a record breaking US$69m. This can be mitigated if each artist has their own public and secret key pair to sign on all their digital assets. An adversary should not be able to find a valid signature of a digital asset that has not been minted by the artist.

Game: Given the artist's public key, can an adversary find a valid signature for an unsigned digital asset?

Π_S =(**SIGN, VRFY**) be a NFT creator authentication scheme based on digital signatures, and consider the forge creator experiment Creator-forge$_{A,\Pi s}$(n):

1. Gen(1^n) is run to obtain the public and secret key pair (pk, sk).
2. Adversary A is given pk and access to an oracle Sign$_{sk}$(·). A then outputs a (D,σ), where D is a data of the digital asset and σ is a signature. Let Q denote the set of all queries that A has asked the oracle.
3. A succeeds if and only if (1) VRFY$_{pk}$(D, σ) = 1 and (2) D ∉ Q. In this case the output of the experiment, Creator-forge$_{A,\Pi s}$(n), is defined to be 1.

The NFT authentication scheme is secure if for all PPT adversaries A, there is a negligible function such that Pr[Creator-forge$_{A,\Pi s}$(n) = 1] ≤ negl(n).

2. Non-Duplicability

The Non-Counterfeitability property prevents an adversary from fraudulent claims that a random digital asset belongs to a certain artist. However, after the artist has minted an NFT of the artist's digital asset, the signature of the digital asset will be public. Without a Non-Duplicability property, an adversary would be able to use the digital signature to mint another NFT token that represents the same digital asset. As such, the NFT scheme should prevent the same digital asset from being minted twice. This can be done if we have a token ID that uniquely identifies the digital asset, and if the smart contract prevents the same token ID from being used more than once (i.e. the same digital asset being minted more than once). However, this token ID will need to be collision resistant, such that no other digital asset can output the same token ID.

Game: Given a token ID which is a hash of the digital asset, can an adversary find another digital asset that outputs the same token ID?

3. Non-Unauthorised Revocability

Non-Duplicability can prevent someone from minting the same digital asset within the same smart contract. However, it is possible for an adversary to create another smart contract and duplicate the same token using all the information that was disclosed in the minting process. For example, an adversary might be motivated to create a duplicate copy of the NFT created by Beeple [10], which was sold for US$69m in March 2021.

To prevent this form of attacks, the NFT should be revocable by the artist via a valid signature. Regardless of whether the artwork was created by Beeple, Beeple certainly did not authorise the artwork. And as such, Beeple should have the power to revoke the NFT and the buyer of the NFT should get back their money (or cryptocurrency that was used to pay for the NFT).

This revocability function should have a time limit in practice, which would act like a holding period for due diligence. However, with this property comes another question of security. Can somebody else other than the artist revoke the NFT?

Game: Given the artists' public key, can an adversary find a valid signature to revoke the NFT?

Similar to the forge creator experiment above, let *REVOKE* = (Sign, VRFY) be a NFT revoke scheme based on digital signatures, and consider the forge revoke experiment $\text{Revoke-forge}_{A,REVOKE}(n)$:

1. $\text{Gen}(1^n)$ is run to obtain the public and secret key pair (pk, sk).
2. Adversary A is given pk and access to an oracle $\text{Sign}_{sk}(\cdot)$. A then outputs a (R,σ), where R is a revoke authorisation and σ is a signature. Let Q denote the set of all queries that A has asked the oracle.
3. A succeeds if and only if (1) $\text{VRFY}_{pk}(R, \sigma) = 1$ and (2) $R \notin Q$. In this case the output of the experiment, $\text{Revoke-forge}_{A,REVOKE}(n)$, is defined to be 1.

The NFT revoke scheme is secure if for all PPT adversaries A, there is a negligible function such that $\Pr[\text{Revoke-forge}_{A,REVOKE}(n) = 1] \leq \text{negl}(n)$.

4. Ownership Security

NFTs are held as tokens in wallets residing in Cryptocurrency Blockchains that support smart contracts such as Ethereum, Solana, Polygon etc. It is critical that the wallet is secure, and that it is not feasible for an adversary to transfer the token away from the wallet. It is common that these Blockchains use a Digital Signature scheme, with the wallet address based on the public key, and that any transactions can only be authorised if it is signed by the corresponding private key. As such, an adversary should be unable to output a valid signature even if it obtains signatures on many other transactions from the same wallet.

Game: Given a wallet address that holds the NFT, can an adversary output a valid signature to transfer the NFT away?

Similar to the forge creator experiment above, let *TRANSFER* = (Sign, VRFY) be a NFT transfer scheme based on digital signatures, and consider the forge transfer experiment $\text{Transfer-forge}_{A,TRANSFER}(n)$:

1. $\text{Gen}(1^n)$ is run to obtain the public and secret key pair (pk, sk).
2. Adversary A is given pk and access to an oracle $\text{Sign}_{sk}(\cdot)$. A then outputs a (T,σ), where T is a transfer instruction and σ is a signature. Let Q denote the set of all queries that A has asked the oracle.
3. A succeeds if and only if (1) $\text{VRFY}_{pk}(T, \sigma) = 1$ and (2) $T \notin Q$. In this case the output of the experiment, $\text{Transfer-forge}_{A,TRANSFER}(n)$, is defined to be 1.

The NFT transfer scheme is secure if for all PPT adversaries A, there is a negligible function such that Pr[Transfer-forge$_{A,TRANSFER}$(n) = 1] ≤ negl(n).

5. Unforgeability

Generally speaking, in terms of the required memory space, it is too expensive to store the entire digital art on cryptocurrency blockchains. Hence, it is more feasible to store a digest instead. To include more information about the digital asset, such as the artist's name and a description of the digital asset, a hash of a metadata which contains such information is used instead of a hash of the digital asset directly. The digital asset will be referenced in the metadata through a hash and a URL which the digital asset will be available.

As such, it is critical that the NFT scheme possesses the property of strong unforgeability, which means that an adversary cannot tamper with the link or hashed digital asset within the minted NFT. Before the digital asset is put into the minting process, SHA256 and Merkle Directed Acyclic Graphs are used to hash it and produce fixed-length hash of the digital asset to be included in the metadata. During another subprocess of minting, hashing is also used to produce fixed-length hashed metadata and digital assets.

Preliminary
Collision-Resistant Hash Function Assumption.
Let H = {H} be a hash family of functions H: {0, 1}* → {0, 1}^{n}. We say that it is Collision-Resistant when an adversary algorithm *A* has the probability in breaking the H if within PPT,
Pr [*A*(n) = (m_0,m_1) : $m_0 \neq m_1$, H (m_0) = H (m_1)] < negl(n)

Game: Given a link between an NFT and a digital art, can an adversary link it to another digital art?

Game Link-forge$_{A,\Pi}$(n): (Forged link finding):

Transaction t_1= (Idx1, l_{Md1}, h_{Md1})
Transaction t_2= (Idx2, l_{Md1}, h_{Md2})
(Idx, l_{Md}, h_{Md} denote tokenId, link to the metadata, hash of the metadata for an digital asset α)

1. Given a digital asset α_1, an adversary algorithm *A* queries its oracle and and trying to a different digital asset α_2 in PPT
2. The adversary wins if and only if $\alpha_1 \neq \alpha_2$, $Idx_1 = Idx_2$, $l_{Md1} = l_{Md2}$, $h_{Md1} = h_{Md2}$

6. Collision Resistant

Unforgeability could be comprehended as preimage resistance which shields the digital asset. Likely, the collision resistance between hashes of metadata could be interpreted as Second preimage resistance, which makes it infeasible to locate a second distinct digital asset with the same output as the given digital asset. Since we are using hashes to represent the metadata and the digital asset, it is critical that the hash functions chosen must be collision resistant, so

that no 2 digital assets will share the same hashes, and no 2 metadata shares the same hash. Otherwise, there will be a lot of confusion as to which metadata and digital asset an NFT is representing.

Game: Given the linking structure between NFTs and digital arts, can an adversary find a pair of digital art that share the same link to an NFT?

Game Link-$coll_{A,\Pi}(n)$: (Link collision finding):

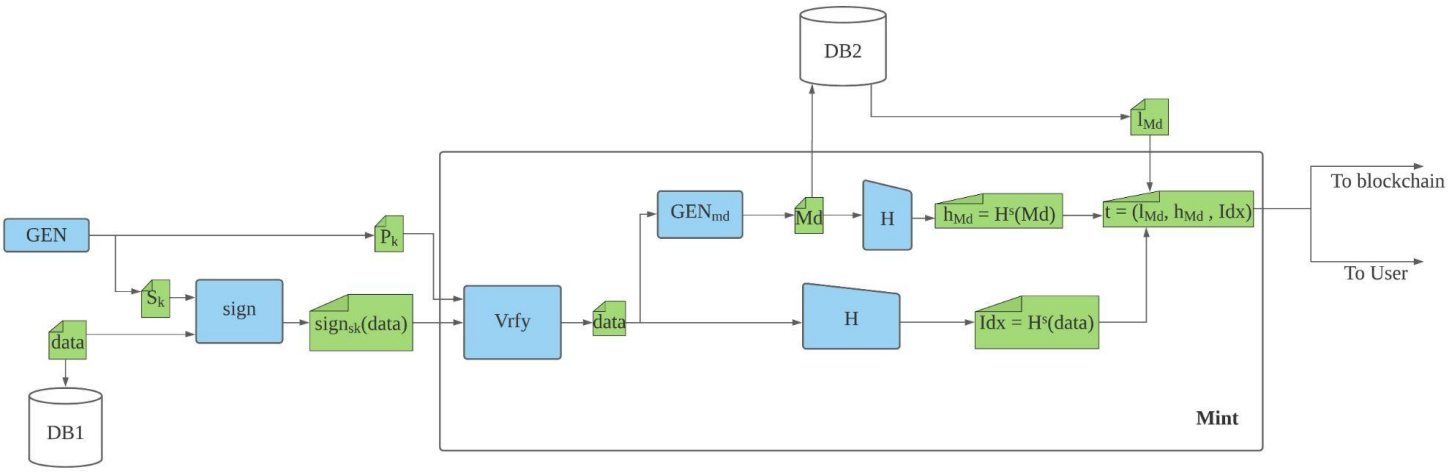
1. An adversary algorithm *A* queries its oracle and and trying to find two different digital asset α_1 and α_2 in PPT
2. The adversary wins if and only if $\alpha_1 \neq \alpha_2$, $Idx_1 = Idx_2$, $l_{Md1} = l_{Md2}$, $h_{Md1} = h_{Md2}$

7. Availability

The NFT's digital asset and the NFT metadata both require an online storage so that they can be publicly available. Otherwise, it would be very difficult for someone to know what the NFT represents. Online storage requires a URL to point to. While we can assume that once these URLs are immutable once they are included in the underlying blockchain, we have to take special care to ensure that the URL continues to point towards the correct files (i.e. the digital asset and the metadata). A distributed storage system will be more robust than a centralised storage system since there will not be a single point of failure. However, for a distributed storage system, there needs to be a consensus on the storage address. Hence, a distributed InterPlanetary File System ("IPFS") that uses a hash of the stored file as the URL would be a good choice to ensure availability, especially since the owner of the NFT would have some degree of control over the availability through sponsoring of nodes within the IPFS ecosystem. This IPFS hash however, will be required to be collision resistant such that no adversary can find another file that carries the same URL and flood the IPFS ecosystem with nodes that serve the adversary's files.

Game: Given a network system, can an adversary make a node unable to fetch a required data from any other nodes?

CONSTRUCTION



Let Π = (**GEN, STORE, MINT, REVOKE, TRANSFER**) be a NFT system for digital assets of arbitrary size, Π_H =(**GEN$_H$**,**H**) be a SHA256 hash function with output length $l(n)$, and let Π_S = (**SIGN, VRFY**) be signature verification based on Elliptic Curve Digital Signature Algorithm (ECDSA). Let there be 3 entities a blockchain β, user u and a interplanetary file system database DB_{ext}. We construct an alternative NFT system Π' = (**GEN', DB1, DB2, MINT', REVOKE', TRANSFER'**) as follows :

- **GEN'** : on input 1^n, run Gen(1^n) to obtain $< P_k, S_k >$ which are the public and private key respectively based on ECDSA.

- **DB1** : takes in some data D where $D \in \{0,1\}^*$ and IPFS database DB_{ext} that is external to the blockchain and invokes $STORE(D, DB_{ext})$ and outputs a url l_D

- **DB2** : takes in the metadata Md of the NFT and IPFS database DB_{ext} and outputs the url l_{Md}, which is based on the IPFS hash of the metadata

- **MINT'** : takes in some data D where $D \in \{0,1\}^*$, the public key P_k of the owner of the data, and the owners digital signature of the data σ where $\sigma \leftarrow SIGN_{S_k}(D)$. The algorithm would then execute the following subprocesses:

1. Verify the signature of the data to ascertain the ownership of the D. On input a public key P_k', Data $D \in \{0,1\}^*$, and a signature σ, Continue the minting process if and only if $VRFY_{P_k}(H\ (D),\ \sigma) = 1$. Stop the minting process otherwise.

2. Generates a token Id Idx where $Idx \leftarrow H\ (D)$[SHA256]

3. Generates a metadata file Md where $Md \leftarrow GEN_{Md}(D)$. The algorithm will then invoke $DB2(Md)$to get l_{Md}, and $H\ (Md)$to get h_{Md} [SHA256]

4. Generates the transaction $t \leftarrow (idx,\ l_{Md},\ h_{Md})$ and is persisted in the blockchain and returned to the user who initiated the minting process

- **REVOKE' :** Takes in a token id Idx, and the users secret key S_k, generates a revoke instruction R, signs the instruction $\sigma \leftarrow sign_{S_k}(R)$and sends $(R,\ \sigma)$ to be persisted in the blockchain. It outputs $VRFY_{P_k}(R,\ \sigma)$

- **TRANSFER' :** Takes in a token id Idx, and the users secret key S_k, generates a transaction T, signs the transaction $\sigma \leftarrow sign_{S_k}(T)$and sends $(T,\ \sigma)$ to be persisted in the blockchain. It outputs $VRFY_{P_k}(T,\ \sigma)$

PROOFS

Our construction is largely based on the security of SHA256 and the Elliptic Curve Digital Signature Algorithm (ECDSA). Both of these algorithms have been in existence for decades, and are used widely in the cryptocurrency world since 2009. Cryptocurrencies have a combined market capitalisation of almost US$3 trillion as of November 2021, which presents itself as a huge honeypot for any adversaries to hack into. Nevertheless, none of them have been successful in breaking SHA256 or ECDSA.

Collision Resistance Proof:

Assume that our system is not collision resistant. This implies that there is an adversary A who can find 2 different input data such that they generate the same t = (l_Md, h_Md , Idx) = (l_{Md}, h_{Md}, H(data)) to be stored in the block chain with success probability greater than ½ + negl(n)

We abstract the construction of the NFT such that it takes in a data and returns a message containing the hash of that data to the user

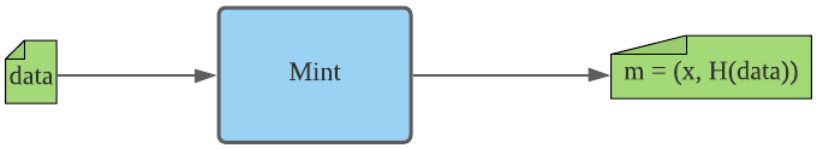


Figure: Abstract construction of NFT

By our assumption, A can find data, data' such that m = m' with an advantage greater than negligible.

We construct A' to solve the collision-finding game of any hash function as follows:

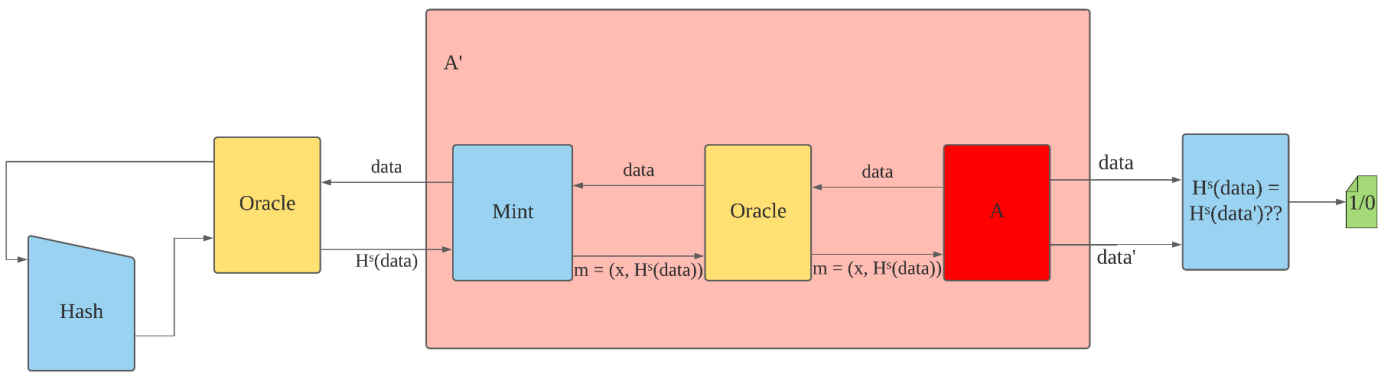


Figure: Construct A' from A

In the diagram above, we customize the mint function to use the outside hashing oracle as a hash function. Since A can find data, data' such that m = m' with an advantage greater than negligible, A' can find data, data' such that $H^s(data) = H^s(data')$ with an advantage greater than negligible. Therefore, by this assumption, hash functions are not collision resistant.

However, since hash functions are collision resistant, the assumption is incorrect, thus the NFT system is collision resistance

Unforgeability Proof: (Target-Collision Resistant)

Assume that our system is not target-collision resistant. This implies that given n, there is an adversary A who can find n' different from n such that they generate the same t = (l_Md, h_Md , Idx) = (l_{Md}, h_{Md}, H(data)) to be stored in the block chain with success probability greater than ½ + negl(n)

We abstract the construction of the NFT such that it takes in a data and returns a message containing the hash of that data to the user

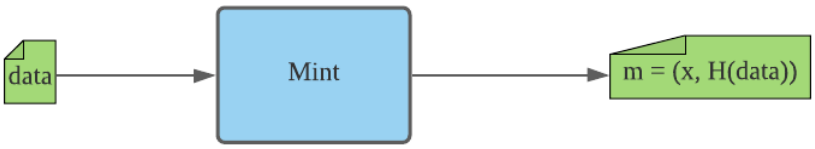


Figure: Abstract construction of NFT

We construct A' to solve the target-collision-finding game of any hash function as follows

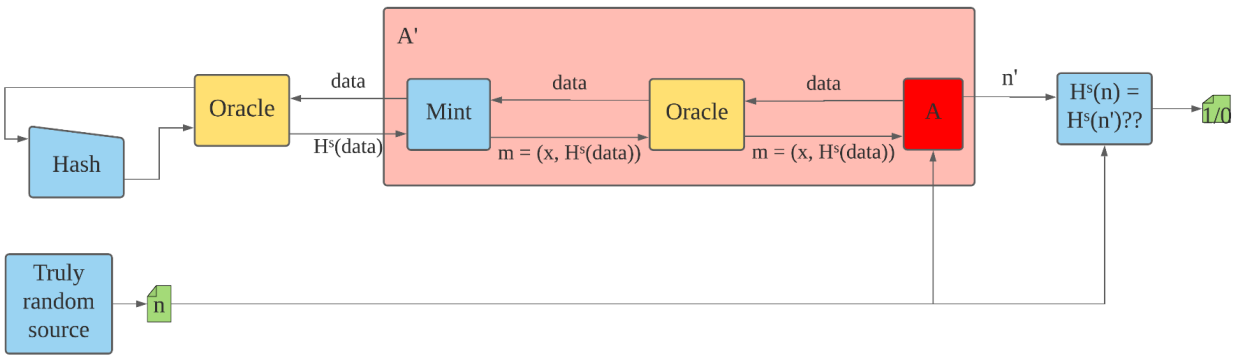


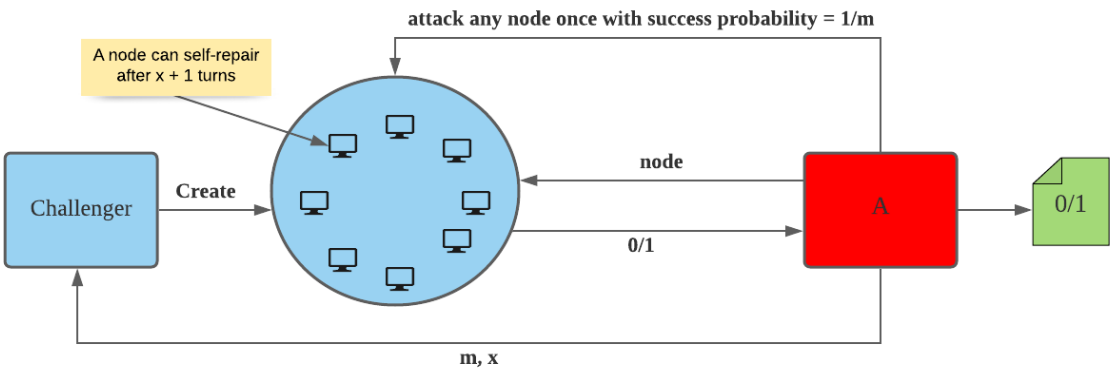
Figure: Construct A' from A

In the diagram above, we customize the mint function to use the outside hashing oracle as a hash function. Since given n, A can find n' != n such that m = m' with an advantage greater than negligible, A' can find n' != n such that $H^s(n) = H^s(n')$ with an advantage greater than negligible. Therefore, by this assumption, hash functions are not target collision resistant.

However, since hash functions are target-collision resistant, the assumption is incorrect, thus the NFT system is target-collision resistant. Therefore, the NFT system is unforgeable.

Availability Proof:

Having the digital assets and their corresponding metadata stored in a distributed IPFS with multiple nodes will provide sufficient levels of availability. Since an IPFS hash is based on Merkel Directed Acyclic Graphs which is target-collision resistant, the only other way an adversary can compromise the availability is to break into every single node.



We want to proof that distributed systems (such as IPFS) provide better availability
We describe the availability game as follows:

Breaking into every single node
Assuming the following:

1. There is a predetermined file F
2. The challenger provides a fully working system with k nodes, each node contains F
3. n = number of independent nodes in the system ≤ k
4. m_i = security parameter of node i, where i = {1,n}
5. m = lowest security parameter of all nodes

Then, each turn, this happens:

- The node: if a node is compromised, it waits for x + 1 turns before fixing itself
- The attacker: every turn, he can either compromise a node (other than i) with a certain success rate of 1/m (m > 1), or require node i to return F.

The adversary wins if node i cannot fetch F from other nodes and return it upon request.
The attacker can choose m, x before sending it to the challenger, the challenger then creates the nodes accordingly based on m and x. We want to prove that the probability that the attacker wins is negligible.

Based on the game, as long as n ≥ x, the only way the attacker can win is to successfully compromise n nodes in n turns. Now, assuming an adversary is able to break each node with a probability of $1/m_i$, which is a very loose assumption, the probability of breaking n nodes in n turns is still less than $1/m^n$, which is negligible no matter what the value of m is. It is more likely that each node has sufficient level of security to achieve negligible probability of being broken, which makes the probability of all nodes even lower. Furthermore, owners of the NFT can sponsor their own nodes to strengthen the availability of their NFT data.

LIMITATIONS

One limitation of our NFT construction is that it is not able to verify whether the creator of the NFT is actually the original creator of the art. An adversary can take an unminted digital asset created by somebody else and be the first person to mint it, under his own name to create the impression that he is the original artist.

Since hashes are used to reference the digital assets, and a change in a single bit of the digital asset can completely change the hashes, an adversary can easily make some insignificant changes to the digital art and mint the NFT with a different token ID which is still valid under our construction.

In practice, this infringes copyright laws. NFT platforms also perform their due diligence, such as reverse image search, to ensure that the digital asset minted is an original not found anywhere else.

CONCLUSION

In comparison to how NFT is implemented in the real world at the time of writing, our NFT construction provides some additional advantages, such as increased availability of the digital assets' data, revocability, non-counterfeitability and non-duplicability. For example, in the record breaking US$69m NFT, the associated artwork is stored on MakersPlace's website. MakerPlace is a startup company that provided the platform for the NFT's sale. If MakersPlace went out of business one day, there is no guarantee that the URL will still work. Also, the token id for the US$69 NFT was arbitrarily chosen, with no direct reference to the associated artwork. This means that anyone can mint another token to reference the same artwork, and even the same metadata. Finally, the US$69m NFT did not provide a means for the original artist to revoke the NFT in the event that his NFT was minted without his authorisation.

We noted that our construction also has limitations, and it will take further research to provide a more complete solution.

REFERENCES

1. *Non-fungible tokens (NFT)*. (n.d.). Ethereum.org. https://ethereum.org/en/nft/
2. Shirole, M., Darisi, M., Bhirud, S.: Cryptocurrency token: An overview. IC-BCT 2019 pp. 133–140 (2020)
3. Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges. *arXiv preprint*, *arXiv:2105.07447*. https://arxiv.org/pdf/2105.07447.pdf
4. *CryptoPunks*. (n.d.). https://www.larvalabs.com/cryptopunks.

5. Zen, A. (2017, Nov 28). *CryptoKitties: The World's First Ethereum Game Launches Today*. Cision. Retrieved 2 Nov, 2021, from https://www.newswire.ca/news-releases/cryptokitties-the-worlds-first-ethereum-game-launches-today-660494073.html
6. Kastrenakes, J. (2021, Mar 11). *Beeple sold an NFT for $69 million*. The Verge. https://www.theverge.com/2021/3/11/22325054/beeple-christies-nft-sale-cost-everydays-69-million
7. Karandikar, N., Chakravorty, A., & Rong, C. (2021). Blockchain Based Transaction System with Fungible and Non-Fungible Tokens for a Community-Based Energy Infrastructure. *sensors 2021*, *21*(3822). https://doi.org/10.3390/s21113822
8. Wang, Q., Li, R., Wang, Q., & Chen, S. (2021). Non-Fungible Token (NFT): Overview, Evaluation, Opportunities and Challenges. *arXiv preprint*, *arXiv:2105.07447*. https://arxiv.org/pdf/2105.07447.pdf
9. Sinclair, C. K. and S. (2021, April 27). *The art of the prank: How A hacker tried to fake the world's most expensive NFT*. CoinDesk Latest Headlines RSS. Retrieved November 5, 2021, from https://www.coindesk.com/markets/2021/04/27/the-art-of-the-prank-how-a-hacker-tried-to-fake-the-worlds-most-expensive-nft/

Cryptographic Study of Robust and Fragile Digital Watermarking Schemes

Mohamed Riyas (A0194608W) John Ng Yee Siang (A0200321J)
Ng Song Guan (A0150099A) Choo Jia Xin (A0206332U)

November 3, 2021

Abstract. Digital watermarking schemes can serve many different purposes related to authenticity and ownership. Depending on their use cases, watermarking schemes may require certain properties and thus schemes are varied. We identify the two main types of watermarking schemes - robust and fragile schemes, and propose that both schemes must possess the properties of unforgeability and unremovability. We clearly define these cryptographic properties through a series of adversarial games. Contrasting the two schemes, we discuss the properties unique to robust and fragile schemes. Robust schemes must possess a stronger property of unremovability while fragile schemes must have the tamper detection property.

Keywords: watermarking, robust, fragile, unforgeability, unremovability

1 Introduction

Protection mechanisms for intellectual property is an important aspect of digital media. Even on just Instagram alone, approximately 95 million photos and videos were being uploaded daily back in 2017. [1] This makes it incredibly difficult for individuals to identify and attribute the original creator of the content. A mechanism that allows the original author of any piece of material to be identified and verified by other individuals and entities would aid in the process of ownership attribution and certain legal cases such as copyright disputes. Digital watermarking is one such mechanism which seeks to achieve the above by embedding or hiding some form of marker into the media work. This work can be an image, audio, video or any other form of transmitted data. This embedded data can then be use to identify the authenticity, integrity and owner of the media, depending on what kind of data was embedded.

2 Robust Watermarking

2.1 Motivation

As making copies of digital media is incredibly easy, digital media is prone to intellectual property rights issues such as copyright infringement and ownership disputes. Robust watermarking seeks to solve this by placing a mark on the media which can identify the original owner. Primarily, the watermark must be able to withstand changes to the underlying media made by an adversary and remain intact despite these changes. These changes include compression, adding noise, filtering, rotation and translation of the media [7]. This allows the owner of the digital media to be correctly identified and prevents false ownership claims.

2.2 Behaviour

The concept of not being able to remove a watermark will be referred to as **unremovability**. Informally, it means that given a data with a robust watermark, an adversary should not be able to create a similar data which does not have the watermark [3]. This thus preserves the ownership of the data.

A robust watermarking scheme should also possess **unforgeability**. An adversary should not be able to create a data (from a previously unmarked data) which is recognised as being marked [3]. This prevents adversaries from attributing false ownership of the data.

3 Fragile Watermarking

3.1 Motivation

With the rise in new technology as well as the mass volumes of digital media being circulated on the Internet, it has become increasingly easier to modify digital media. There are also issues surrounding intellectual property rights as there are cases of false claims of digital media ownership [9]. Hence, there is a need for schemes to protect the integrity and authenticity of digital media [10]. Fragile watermarking is one of such schemes that can ensure the integrity and authentication of digital media.

3.2 Behaviour

Fragile watermarks are designed to detect minute changes in digital media. Unlike robust watermarking, a slight change in the watermarked digital media would destroy the fragile watermark [6], ensuring the integrity of the digital media. This property of fragile watermarking makes it ideal for digital media **modification** or **tampering detection** [8].

Fragile watermarking is similar to digital signatures in the sense that both provide authentication. However, fragile watermarking has some advantages over digital signatures. For example, digital signatures are explicitly attached to the data while fragile watermarks are discretely embedded directly in the data [2]. This makes fragile watermarks harder to remove than digital signatures. The sensitive nature of fragile watermarking also ensures the authenticity of digital media as it makes it difficult for attackers to falsely claim ownership of the digital media[4].

4 Watermarking Overview

4.1 Construction of Watermarking Model

In this section, we will be presenting a construction of a formal generic key-based watermarking model. In our model, we would be defining the necessary inputs, outputs and component functions of the watermarking scheme.

One challenge here is whether to consider the 'complete' set of watermarking inputs, outputs, and component functions from all their specific information domains and function families. To reduce the complexity, we will take the reduced approach of defining a set of 'possible' inputs, outputs and component functions that can capture the fundamental properties of well-known schemes used today.

Irrespective of the system and security requirements, a watermarking scheme usually has three fundamental components. We term them as *Gen*, $Mark$ and *Detect* in our model, as shown in Figure 1. The primary roles of these functions in a typical watermarking application, the inputs they take in and the outputs they produce will all be discussed below and has also been summarized in Figure 1.

Gen Function This function is responsible for generating a key given some number, n. This key will be later used in the $Mark$ function to generate a watermark and also in the *Detect* function to verify a watermark.

Mark Function This function is responsible for first generating a suitable watermark and then embedding the watermark into the data that needs to be watermarked. We will first briefly discuss about the generation of the watermark.

In any application, it is important that watermarks are generated depending upon the required watermark objectives in that particular application. For instance, as discussed in section 2, in a copyright protection application, a watermark may need to be robust to resist certain processing techniques and/or malicious attacks. If these requirements are not considered, that can lead to security vulnerabilities and technical flaws.

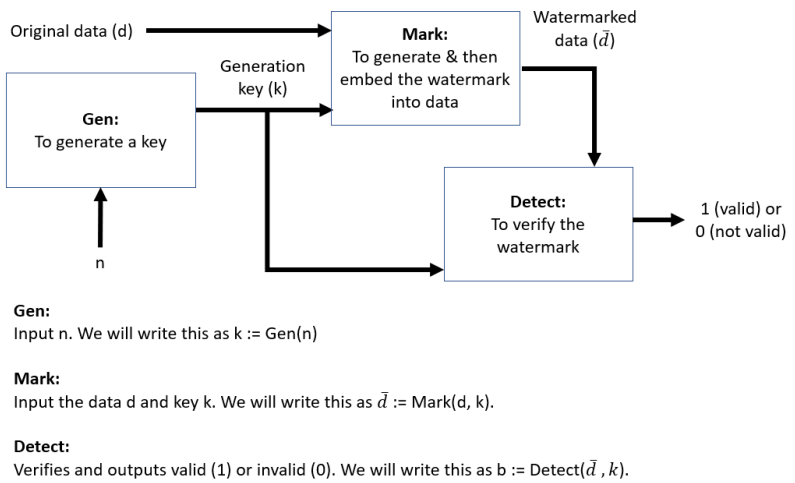


Fig. 1. Construction of game models

Moving on to watermark embedding, the $Mark$ function considers where and how to embed the watermark such that the certain application requirements are satisfied. In addition, there are different types of embedding (e.g: invisible, invertible, reversible, etc) and different domains (e.g., spatial, transform) for embedding. We will not discuss them in this report to present a simplified model for better ease of understanding.

With that being said, irrespective of the constraints, embedding domain and type, the $Mark$ function works by first taking in the original data that needs to watermarked, d (e.g. image data for an image application) and a generation key, k, as inputs. The function then computes a watermark, w and embeds it into the d to output the watermarked data, $\bar{d}$.

Detect Function This function is responsible for making an objective decision whether the watermarked data is valid. The basic idea is that the function first extracts the embedded watermark and regenerates another version of the watermark from the inputs. If both the regenerated and the extracted watermark matches each other, the function would output 1, to indicate that the watermarked data is valid. Otherwise, the function outputs 0. Similarly, if the function is unable to detect and extract the watermark from the watermarked data, it also outputs 0. Like the $Mark$ function, the internal design of the $Detect$ function can also vary. For instance, the $Detect$ function may also be tweaked to extract the embedded data and output it as the estimated data if a valid watermark is detected by the function. Irrespective of its internal design, the

Detect function generally takes in the watermarked data $\bar{d}$, and the generation key, k to yield either 1 or 0 depending on whether the watermarked data is valid.

4.2 Other variations

In our model, for simplicity, the $Mark$ function is considered to generate and embed the watermark. However, this can naturally be split into separate functions as well. Also, in our model, we consider the original data as input for the watermarking functions. However, there may be cases where a valid watermarked version of the data can be used as the input. For instance, to update/re-embed a watermark in an existing watermarked image, the watermarked image may need to be used as the input rather than the original image.

4.3 Game Constructs

So far, we have seen the construction of a formal generic watermarking model as a key-based model. Moving on, we will be utilizing the model to define attack models and for carrying out other related analysis of a watermarking scheme.

Marking Oracle $\bar{d} := MO(d)$
Given a data d, marking oracle MO will return $\bar{d}$ such that $\bar{d} := Mark(d, k)$. We assume k is already supplied to the marking oracle and MO only takes d as input.

Detection Oracle $b := DO(d')$ such that $b = 1$ or 0
Given a data d', detection oracle DO will return 1 iff $1 := Detect(d', k)$ and will return 0 iff $0 := Detect(d', k)$. We assume k is already supplied to the detection oracle and DO only takes d' as input.

5 Desirable properties of watermarking schemes

5.1 Unforgeability

An important property for all watermarking scheme is that of unforgeability. Intuitively, we want it to be extremely difficult for an adversary to construct a data that is recognised by the detector as being marked, ie. difficult to construct a data d' such that $Detect(d', k)$ returns 1. In the context of digital media and products, if a watermark can be forged, it would be impossible to tell whether a particular product is genuine or authentic.

Unforgeability for Fragile Schemes For fragile watermarking schemes, the concept of unforgeability is very similar to that of existential unforgeability of a Secure MAC, which is expected since the two are similar in function. In the following game, we consider unforgeability under an adaptive chosen-data attack, where the adversary has access to a marking oracle.

Unforgeability Game for Fragile Schemes $Fragile - forge_{\mathcal{A},\sqcap}(n)$

1. Generate key $k := Gen(n)$
2. Adversary has marking oracle access. Marking oracle takes in a data d and returns the marked data $\bar{d} := Mark(d, k)$. Adversary outputs d'. Let Q be all the $\bar{d}$ the adversary has obtained from the marking oracle in this experiment.
3. The adversary wins if and only if $Detect(d', k)$ returns 1 (true) and $d' \notin Q$.

Unforgeability Definition for Fragile Schemes A fragile watermarking scheme $\sqcap$ is said to be unforgeable under an adaptive chosen-data attack if and only if

$$Pr(Fragile - forge_{\mathcal{A},\sqcap}(n) = 1) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully forge a valid marked data.

Unforgeability for Robust Schemes A robust scheme requires a slightly modified game due to the stronger unremovability property it possesses. This means given a data marked by a robust scheme $\bar{d}$, if we modify $\bar{d}$ to d' such that $\bar{d}$ and d' are still similar, d' remains detectable, ie. $Detect(d', k)$ returns 1. Hence, we have to modify step 3 of the unforgeability game for fragile schemes to accommodate this property.

Unforgeability Game for Robust Schemes $Robust - forge_{\mathcal{A},\sqcap}(n)$

1. Generate key $k := Gen(n)$
2. Adversary has marking oracle access. Marking oracle takes in a data d and returns the marked data $\bar{d} := Mark(d, k)$. Adversary outputs d'. Let Q be all the $\bar{d}$ the adversary has obtained from the marking oracle in this experiment.
3. The adversary wins if and only if $Detect(d', k)$ returns 1 (true) and d' is not similar to any data in Q.

Unforgeability Definition for Robust Schemes A robust watermarking scheme $\sqcap$ is said to be unforgeable under an adaptive chosen-data attack if and only if

$$Pr(Robust - forge_{\mathcal{A},\sqcap}(n) = 1) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully forge a valid marked data.

5.2 Tamper Detection

Tamper Detection applies to watermarks that are fragile. When a data that has been watermarked is subsequently modified, the watermark should be rendered invalid. To defeat tamper detection, an adversary would attempt to modify the watermarked data while preserving the validity of the watermark. This makes

it more difficult than the forging attack as the initial data is fixed, rather than decided by the attacker. One practical application of tamper resistance is in digital contracts. It can be used to distinguish which version of the contract was signed by both parties. Should anyone attempt to tamper the contract after it was signed and watermarked, the watermarking would break indicating that the contract has been tampered with.

Tamper Detection Game, $Tamper_{\mathcal{A},\sqcap}(n)$

1. Generate challenge data d and key k, $k := Gen(n)$ and challenge data $d = \{0,1\}^n$.
2. Produce watermarked data $\bar{d} := Mark(d,k)$ and give it to the adversary
3. The adversary analyzes $\bar{d}$ and produces d' which is similar to $\bar{d}$ but $\bar{d} \neq d'$
4. The adversary wins iff $Detect(d',k)$ returns 1 (true)

Tamper Detection Definition (Fragile Schemes) A watermarking scheme $\sqcap$ is said to have tamper detection if and only if

$$Pr(Tamper_{\mathcal{A},\sqcap}(n) = 1) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully tamper any given piece of data while preserving the watermark. This concept is foundational to the idea of a fragile watermarking scheme, as it concerns the authenticity of any given watermarked document and provides non-repudiation.

Tamper Resistance for Robust Schemes Due to the way that robust watermarking schemes work, the interaction with tampering functions slightly differently. A robust watermark will preserve itself under modification. This means that all robust watermarking schemes do not possess tamper detection. Therefore, in the case of our above definition, the chance of winning the game under any good robust scheme $\sqcap$ evaluates to the following expression.

$$Pr(Tamper_{\mathcal{A},\sqcap}(n) = 1) \geq 1 - negl(n)$$

Hence for robust schemes we consider tamper resistance to be its ability to preserve the validity of the watermark upon modification. This definition highlights the fundamental difference between fragile and robust schemes. We will formally discuss other ways to define this in greater detail the subsequent sections about the property of unremovability.

Tamper Detection versus Unforgeability In this section we discuss the relationship between tamper detection and unforgeability for fragile schemes. We will prove that unforgeability is a necessary condition for tamper detection as long as the scheme is fragile by proving the contrapositive where all schemes that do not exhibit tamper detection are not unforgeable.

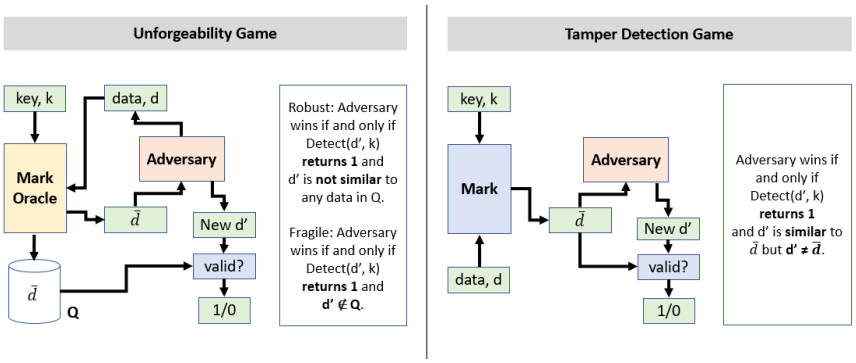


Fig. 2. Construction of the Unforgeability and the Tamper Detection games

Proof. 1. Consider a fragile watermarking scheme $\sqcap$ that does not exhibit tamper detection
2. This implies that the adversary can construct a valid watermark object by modifying another valid watermarked object
3. In order to win the unforgeability game, the adversary can query the watermarking oracle with any message d to produce $\bar{d}$
4. Since the scheme lacks tamper detection, the adversary modifies $\bar{d}$ to obtain d' which is still a valid watermark but $\bar{d} \neq d'$
5. The adversary submits d' which is accepted as $d' \notin Q$ and wins.
6. Therefore, the scheme $\sqcap$ also does not exhibit unforgeability.

However, does tamper detection imply unforgeability?

Proof. 1. Consider a fragile watermarking scheme $\sqcap$ that exhibits tamper detection
2. This implies the adversary cannot successfully produce any valid watermarked data similar to the challenge data
3. Using the watermarking oracle, the adversary can generate any arbitrary watermarked object d
4. Since $\sqcap$ exhibits tamper detection, modification of d to d' invalidates the watermark.
5. Thus it is impossible to use the watermarking oracle to generate a new valid watermarked object via modification.
6. However, there is insufficient evidence to show that a valid watermark cannot be generated by a polytime adversary through other means, such as through use of statistical analysis over a large set of watermarked objects.

As we've established, unforgeability is a necessary condition for tamper detection. However, the reverse is not true. Tamper detection alone is insufficient to assert that the scheme is unforgeable.

5.3 Unremovability

Unremovability is an important property of watermarks as it is the foundation in which authentication relies on. If a watermark is easily removed, protection of intellectual property fails under watermarking schemes as there is no longer a way to authenticate the data. Hence, it is vital for both Robust and Fragile watermarking schemes to exhibit unremovability.

For a watermarking scheme to be effective in preserving the authenticity of digital media, it should be difficult for an adversary to create a data that is similar to a watermarked image, but no watermark is detected. In other words, an adversary should not be able to remove the watermark from a watermarked data. Removing a watermark does not necessarily entail reproducing the data that is completely identical to the original unmarked data. For robust watermarking schemes, if an adversary is able to produce a data that is 'similar' to the watermarked data such that no watermark is detected, the watermark is considered to be removed. The term, 'similar' is ambiguous and may vary with the application scenario as different applications might have different methods of calculating similarity and defining thresholds to determine whether a given set of data can be termed as similar. In this paper, since similarity is not the focus of the paper, we will not be discussing about them further.

The following games discussed below are for robust watermarking schemes.

Unremovability Game for Robust Schemes, $\boldsymbol{Remove_{\mathcal{A},\sqcap}(n)}$ In this game, given a watermarked data, the aim of the adversary is to output a data that is 'similar' to the given watermarked data but at the same time, will never be detected as containing the watermark. Note that the adversary will also win the game if he manages to output a data that is completely identical to the original unmarked data. However, for simplicity, we will treat it as the output data being 'similar' to the given watermarked data. The game can be constructed as below:

1. Generate challenge data d and key k, $k := Gen(n)$.
2. Produce watermarked data $\bar{d} := Mark(d, k)$ and give d to the adversary
3. The adversary analyzes d and produces d' such that d' is similar to d
4. The adversary wins if $Detect(d', k)$ returns 0 (false)

Here, for a stronger adversary, the input can also include the watermark used to embed the data,d and the adversary can have access to MO $(\cdot)$.

Unremovability Definition for Robust Schemes A robust watermarking scheme $\sqcap$ is said to be unremovable if and only if

$$Pr(Remove_{\mathcal{A},\sqcap}(n) = 0) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully construct a data that is 'similar' to the given watermarked data but at the same time, will never be detected as containing the watermark.

Strong Unremovability Game for Robust Schemes, $sRemove_{\mathcal{A},\sqcap}(n)$ In the previous experiment, the adversary only has access to a single watermarked data. However, it is possible for an adversary to have access to multiple copies of a watermarked data. In cases where the watermarking scheme is probabilistic, it is also possible to obtain multiple copies of a watermarked data, each with a different watermark.

This motivates the definition of a strongly unremovable watermarking scheme. The game can be constructed as below:

1. Generate challenge data d and key k, $k := Gen(n)$.
2. Let Q be all the copies of watermarked data produced, such that $\forall \bar{d} \in Q, \bar{d} := Mark(d, k)$. Give Q to the adversary.
3. The adversary analyzes Q and produces d' such that $\exists c \in Q$ that d' is similar to.
4. The adversary wins if $Detect(d', k)$ returns 0 (false)

Strong Unremovability Definition for Robust Schemes A robust watermarking scheme $\sqcap$ is said to be strongly unremovable if and only if

$$Pr(sRemove_{\mathcal{A},\sqcap}(n) = 0) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully construct a data that is 'similar' to any one of the given copies of a watermarked data but at the same time, will never be detected as containing the watermark.

Unremovability for Fragile Schemes As fragile schemes are used for tamper detection, this means that any small modification to a watermarked data $\bar{d}$ to produce a similar data d' would result in $Detect(d', k)$ returning 0 (false). Hence, under both the unremovability and strong unremovability game for robust schemes, the adversary will always win if the watermarking scheme used is a fragile one.

This motivates a stricter criteria for unremovability for fragile schemes. As such, instead of producing a data similar to the original watermarked data, the adversary only wins the game if it is able to produce a data d' that is completely identical to the original unmarked data d. Note that the stricter criteria makes it harder for the adversary to win the unremovability game. Hence, robust schemes have a stronger notion of security for unremovability compared to fragile schemes.

Unremovability games for fragile schemes are therefore the same as robust schemes, except for the winning criteria. For example, the following game is the same as before, with the winning criteria modified to be more strict.

1. Generate challenge data d and key k, $k := Gen(n)$.
2. Produce watermarked data $\bar{d} := Mark(d, k)$ and give $\bar{d}$ to the adversary
3. The adversary analyzes $\bar{d}$ and produces d'
4. The adversary wins if d' is exactly equal to d (output 1)

Unremovability Definitions for Fragile Schemes Given the change in winning criteria for fragile schemes, the definition of unremovability and strong unremovability is slightly modified as well.

As the adversary now wins when output of the game is 1 instead of 0, a fragile watermarking scheme $\sqcap$ is said to be unremovable if and only if

$$Pr(Remove_{\mathcal{A},\sqcap}(n) = 1) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully construct a data that is completely identical to the original unmarked data.

Similarly, a fragile watermarking scheme $\sqcap$ is said to be strongly unremovable if and only if

$$Pr(sRemove_{\mathcal{A},\sqcap}(n) = 1) \leq negl(n)$$

This means that against any polytime adversary $\mathcal{A}$ there is a negligible chance that they are able to successfully construct a data that is completely identical to the original unmarked data.

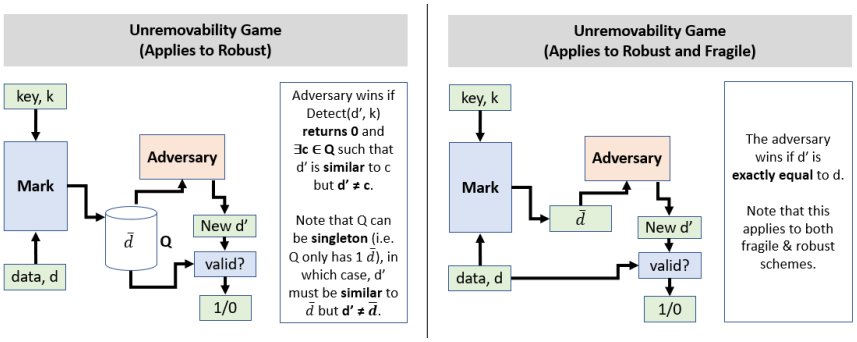


Fig. 3. Construction of the Unremovability games

6 Conclusion

We have discussed the key properties of all watermarking schemes which are unforgeability and unremovability. In summary, unforgeability means an adversary is unable to produce marked data from unmarked data; unremovability means an adversary is unable to produce the original unmarked data from marked data. An adversary which can succeed in either of the attacks can render a watermarking scheme useless. We have also shown the difference between robust and fragile schemes. In particular, robust schemes require a stronger notion of unremovability, where the adversary is unable to even produce unmarked data which is similar to the marked data. For fragile schemes, we identified the tamper detection property where given an original marked data, an adversary is unable to produce modified data which is marked yet similar to original data.

References

1. 31 mind-boggling Instagram Stats & Facts for 2022. WordStream. (n.d.). Retrieved November 3, 2021, from *https* : *//www.wordstream.com/blog/ws/*2017/04/20/*instagram − statistics*
2. Celik, Mehmet & Sharma, Gaurav & Saber, Eli & Tekalp, A.. (2002). Hierarchical watermarking for secure image authentication with localization. IEEE Transactions on Image Processing. 11. p. 585-595. 10.1109/TIP.2002.1014990.
3. Cohen, A., Holmgren, J., & Vaikuntanathan, V. (2015). Publicly Verifiable Software Watermarking. IACR Cryptol. ePrint Arch., 2015, 373.
4. Eugene T. Lin and Edward J. Delp. (2001). A Review of Fragile Image Watermarks, p. 2. *https* : *//www.cerias.purdue.edu/assets/pdf/bibtex*$_a$*rchive/*2001 − 138 − *report.pdf*
5. Hopper N., Molnar D., Wagner D. (2007) From Weak to Strong Watermarking. In: Vadhan S.P. (eds) Theory of Cryptography. TCC 2007. Lecture Notes in Computer Science, vol 4392. Springer, Berlin, Heidelberg. *https* : *//doi.org/*10.1007/978 − 3 − 540 − 70936 − 7$_2$0
6. Kiran, Kanwar Garg. (2015). Digital Watermarking: Potential Challenges and Issues, p. 2. *http* : *//www.ijcset.net/docs/Volumes/volume*5*issue*3/*ijcset*2015050304*.pdf*
7. Lian, S. (2008). Multimedia Encryption and Watermarking in Wireless Environment. Handbook of Research on Wireless Security, 236–255. doi:10.4018/978-1-59904-899-4.ch016
8. Prasad, S., Pal, A.K. (2020). A Secure Fragile Watermarking Scheme for Protecting Integrity of Digital Images. Iran J Sci Technol Trans Electr Eng 44, p. 703–727. https://doi.org/10.1007/s40998-019-00275-7
9. Sonali V. Satonkar, Dr. Seema Kawathekar. (2013). Overview of Watermarks, Fingerprints, and Digital Signatures. INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 02, Issue 12 (December 2013).
10. Thanki, R., Borra, S. (2019). Fragile watermarking for copyright authentication and tamper detection of medical images using compressive sensing (CS) based encryption and contourlet domain processing. Multimed Tools Appl 78, p. 13905–13924. *https* : *//doi.org/*10.1007/*s*11042 − 018 − 6746 − 2

Non-Fungible Tokens: New-age Profits? Or just a gimmick.

Joshua Tan Yin Feng[A0199502Y], Wu Yifan[A0242690E], Foong Yan Kai, Brandon [A0135261N], and Donavan Lim Jia Hui[A0206088A]

National University of Singapore
joshua.tyf@u.nus.edu
yifan.wu@u.nus.edu
brandon.foong@u.nus.edu
donavanlim@u.nus.edu

Abstract. Non-fungible tokens (NFTs) are transferrable, unique, and non-duplicatable units of data registered on a blockchain digital ledger. NFTs represent rights to digital assets such as digital art and in-game collectables. Despite the zealous adoption and unrelenting trust shown by the NFT user base towards the safety of exchanging NFTs for economic gains, recent cases of users being deceived or losing their cryptocurrency due to social-engineering tactics have raised salient risks on using NFTs. We investigate two broad security implementations that underpin the implementations used to mint NFTs on third-party marketplaces: a mint-impersonation scheme and a lottery scheme. By devising schemes, constructions, and proofs, we aim to demonstrate how analysts can qualify the security of such NFT-based implementations, and as such realizing the relevance and gravitas of these security qualities in a system that holds the monetary stake of an expanding user base.

Keywords: Non-Fungible Tokens; Cryptography; Cryptocurrency.

1 Introduction

1.1 The Fervid Adoption of NFTs

Non-fungible tokens (NFTs) have seen a rise in popularity in recent times. NFTs can be considered a proof of ownership of a digital asset, a much-needed feature in today's highly digitalised world. Prior to NFTs, it was difficult for anyone to verify their ownership over digital assets since exact duplicates could be made without difficulty. Premised on the blockchain, NFTs enable verifiable ownership by associating digital assets to a digital blockchain wallet [1]. Association could be as simple as storing the pair of unique ids of an asset and wallet on the blockchain. The owner of an NFT can easily prove his ownership by the private key that gives access to the digital wallet. Coupled with the immutable property of the blockchain, these associations are safe from malicious modifications and hence protect the ownership rights of an NFT [7].

NFTs are a great solution to introduce the concept of scarcity in a digital world. Scarcity promotes the appraisal of digital assets and successfully creates a marketplace for them. Instead of trading these assets directly, NFTs are traded on their behalf. The non-fungibility of NFTs means that they are unequal in value and cannot be interchanged with each other, unlike fiat currency or cryptocurrency [9]. To trade NFTs, traditional cryptocurrencies such as Ethereum are often used in NFT-exchange on third-party marketplace platforms [1].

Investors have hailed NFTs as a potentially disruptive innovation that is paving the way for wealth preservation through digital asset ownership [12]. As such, NFTs have experienced a meteoric rise in sales volume and attention since its inception in 2015, with increasing traction from the digital art and gaming industries. NFT-exchange skyrocketed from 1.8 billion USD in Q2 2021 to 10.7 billion USD in Q3 2021, signifying the volumetric adoption of NFTs mostly in the domain of personal and non-commercial use [11].

It is important to note that an NFT, however, does not confer copyright protection over the digital asset it represents. As such, a buyer of an NFT merely obtains a license to use the art piece and cannot prevent the original author from creating more NFTs of the same work. In short, the owner of an NFT does not own nor have the right to the actual digital asset the NFT represents; the owner instead owns a certificate of authenticity, often represented by a hash code that can be exchanged on marketplaces [4].

1.2 Identifying Insecurities of NFTs

While many adopt a positive outlook for NFTs as the future of digital assets amidst fast adoption, issues pertaining to security that leave users vulnerable exist.

NFTs are not a standalone technology. Instead, they are built upon the blockchain, smart contracts, the internet, etc. Security of NFTs is thus dependent on these technologies as well. With digital signatures and hash functions underscoring the blockchain, transactions made on it can be proven to be secure. However, the security of the blockchain is only a *necessary* condition for the security of NFTs, not a *sufficient* one. There have been several instances of artists who had discovered their works sold on NFT platforms without their consent, breaching copyright. Yet, these transactions do not contradict the security of the blockchain. The blockchain is only concerned with ensuring the verifiability of the person making the transaction and the immutability of its entire transaction history. In the former case, a thief selling a piece of art not belonging to him does not contradict the verifiability of the person making the transaction.

Even though platforms have taken measures to curb fraud, many scammers still successfully deceive users to believe that they are the original creator, and illegally resell artists' works on their behalf to profit off their credibility and reputation [3]. On the other hand, there also exists the issue of theft, where hackers manipulate people to give up their account information via ransomware or use malware to steal their accounts. Once their accounts are compromised, the victims' NFTs in them are also

stolen or resold for profits [2]. All these issues further motivate the importance of creating secure schemes for NFTs.

2 The Mint-Impersonation Scheme

2.1 Impersonating an NFT mint

As alluded to in the previous section, fraud can be committed when NFTs for digital assets under an artist's name can be minted without their consent and then sold on the marketplace. The profits are pocketed into the accounts of the fraudster who minted without the artist's consent. This tactic is known as sleepminting [5].

2.2 The ERC721 Standard

NFTs are created through smart contracts, which are essentially just pieces of code that dictate the actions that can be done with regards to the NFTs. As a form of standardisation, the NFT community collectively decided on a standard known as ERC721 [6]. Being an interface (in the object-oriented programming sense), ERC721 is simply just a set functions that the NFT smart contracts must have. This standard specifies how the tracking and transacting of NFTs created by the smart contract must be done.

ERC721 however does not specify secure or good implementations for minting and burning (destroying) NFTs [6]. Furthermore, there is no security provided by the ERC721 standard itself. Being just mere pieces of code, the smart contract can be modified such that it still follows the ERC721 standard and yet allow for fraudulent mints. This is possible because NFT marketplace service providers, such as *OpenSea* or *Rarible*, do not dictate the full implementation of smart contracts based on the ERC721 standard. Finlow-Bates [8] provides a concrete example of how this could be achieved. Here, a fraudster can create an insecure smart contact that provides backdoor access to the NFTs minted into other accounts. Afterwards, using the backdoor access, they can approve transactions and claim profits despite not being the rightful owners of these NFTs.

This underscores the relevance of understanding the minting process of an NFT and creating secure minting schemes to protect users from fraud.

2.3 Mint Scheme

In this section, we introduce an authenticated minting scheme. The idea is that we want a way to prove that any person intending to lay claims on an NFT, is indeed the rightful owner.

Definition 1.1 Mint: *A* $Mint$ *consists of 3 probabilistic polynomial-time algorithms* $(Gen, AssociateOwner, VerifyOwner)$ *such that*:

- *The key-generation algorithm* Gen *takes as input the security parameter* 1^n *and outputs a random key* k *and a random string* r*, with* $|k|, |r| \geq n$
- *The owner-association algorithm* $AssociateOwner$ *takes as inputs secret key* k *as well as strings* $itemId$ *and* r*, and outputs the string* tag.
- *The owner-verification algorithm* $VerifyOwner$ *takes as inputs key* k *as well as strings* $itemId$*,* r *and* tag*. It then outputs a bit* b *with* $b = 1$ *meaning valid and* $b = 0$ *meaning invalid.*

It is required that for every n, every (k, r) generated by $Gen(1^n)$, and every $itemId$, it holds that $VerifyOwner_k(itemId, r, tag) = 1$, where tag is generated from $AssociateOwner_k(itemId, r)$.

To model after the scheme, we create a mint impersonation game. The intuition and inspiration behind it are briefly described as follows: prior to minting, a message authentication code tag, corresponding to the asset whose NFT is to be minted, would be first created using the key k belonging to the owner of the asset. The user is given the $itemId$ of the asset and a random value r. $itemId$ serves as a representation of the asset and the tag will serve as an irrevocable proof of the original owner. Both $itemId$ and tag will be included publicly in the blockchain during the minting process. After successful minting, the NFT should be safe from an adversary attempting to claim that they are the owner. Verification of the NFT would involve the adversary producing a tag and checking the validity of it. Logically, the tag must belong to an asset that has been minted. The difficulty lies in creating a valid tag without having knowledge of the key k and random value r.

More formally, we define the mint impersonation game with adversary A, a security parameter n and a mint scheme $\Pi = (Gen, AssociateOwner, VerifyOwner)$ as follows:

The mint impersonation game $Mint - Impersonate_{A,\Pi}(n)$**:**

1. The owner's key k and a random value r are generated by running $Gen(1^n)$.
2. The adversary A is given oracle access to $AssociateOwner_{k,r}(\cdot)$.
3. A can query its oracle $AssociateOwner_{k,r}$ with any string $itemId$ of its choice. With each query, the oracle will produce and return the value tag to A.
4. Oracle also stores the i^{th} tag associated with the $itemId$ in its i^{th} query in a set Q as a pair $< itemId, tag >$.
5. A is allowed to query its oracle $AssociateOwner_k$ as many times as it wants.
6. Eventually, A outputs the values $< itemId', r', tag' >$
7. A succeeds if, and only if:
    a. $VerifyOwner_k(itemId', r', tag') = 1$; and
    b. the pair $< itemId', tag' > \in Q$.
8. On success, the output of the experiment is defined to be 1. Else, the output is 0.

A Mint is non-impersonate-able if no efficient adversary can succeed in the above experiment with non-negligible probability.

Definition 1.2: *A mint scheme* $\Pi = (\text{Gen}, \text{AssociateOwner}, \text{VerifyOwner})$ *is non-impersonate-able if, and only if, for any probabilistic polynomial time adversary* A*, there exists a negligible function* $negl$ *such that:*

$$\Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1\right] \leq negl(n)$$

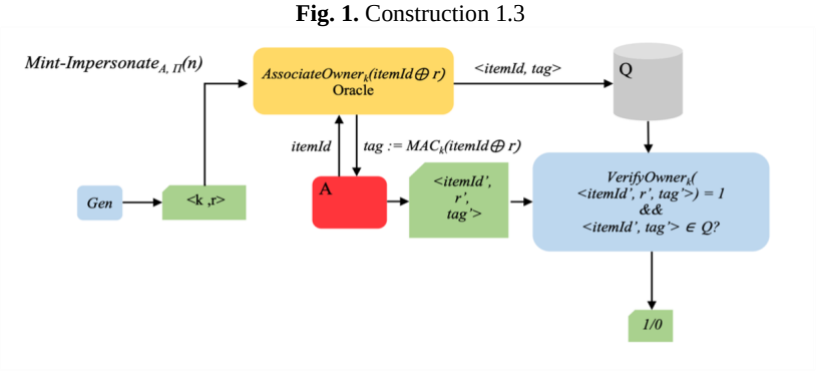
2.4 Constructing a secure Mint scheme

Our construction revolves around the usage of a secure MAC scheme. Intuitively, we can use a MAC to create the tags; if the MAC is secure, then it should not be possible for any efficient adversary to create another valid tag.

Construction 1.3: Let $\Pi' = (Gen, Mac, Vrfy)$ be a secure MAC scheme. We define a $Mint$ with security parameter n as follows:

- Gen: On input 1^n, run $Gen(1^n)$ to obtain (k, r), with key $k \in \{0,1\}^n$ and random value $r \in \{0,1\}^n$.
- $AssociateOwner$: On input $(itemId, r)$, output $tag \coloneqq Mac_k(itemId \oplus r)$.
- $VerifyOwner$: On input $(itemId, r, tag)$, return $Vrfy_k(itemId \oplus r, tag)$.

Fig. 1. Construction 1.3



Theorem 1.4: If Π' is a secure MAC scheme, then Construction 1.3 is *non-impersonate-able*.

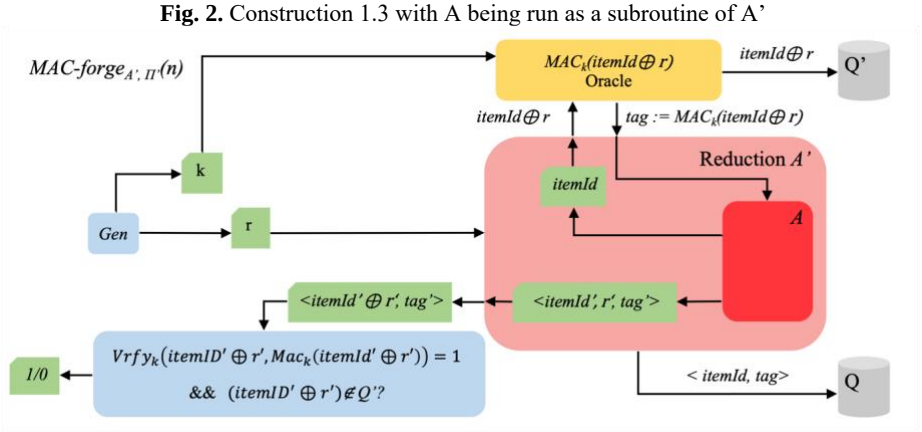
Proof: We seek to prove the above theorem by contradiction. Let Π be our construction. To aid our proof, we construct another adversary A' that seeks to create a valid forge on the MAC scheme Π'. A' runs A as a subroutine and works as follows:

1. Run $Gen(1^n)$ to create random value r and secret key k
2. A' is given r and has access to an oracle $Mac_k(\cdot)$
3. A' runs A and serves as its oracle $AssociateOwner$. A will query A' with $itemId$. On every query that A makes, answer it as follows:
4. Query $Mac_k(itemId \oplus r)$ and get tag back in return

5. Return $tag\ to\ A$
6. Eventually, A outputs $(itemId', r', tag')$
7. A' then outputs $(itemId' \oplus r', tag')$. The forge experiment on Π' outputs 1 if 1) $itemId' \oplus r'$ and tag' is valid, and 2) $itemId' \oplus r' \notin Q'$ where Q' represents the set of all messages queried on $Mac_k(\cdot)$

One can see that A' is emulating Π for A.

Fig. 2. Construction 1.3 with A being run as a subroutine of A'



By the law of total probability,

$$\begin{aligned}&\Pr[Mint - Impersonate_{A,\Pi}(n) = 1]\\ &= \Pr[Mint - Impersonate_{A,\Pi}(n) = 1 | r' = r] \cdot \Pr[r' = r]\\ &\qquad + \Pr[Mint - Impersonate_{A,\Pi}(n) = 1 | r' \neq r] \cdot \Pr[r' \neq r]\end{aligned}$$

Case 1: $r' = r$

From the definition of $Mint - Impersonate_{A,\Pi}$, the adversary will win with probability 1 if he can output $itemId'$, r' and tag' such that $r' = r$ and $< itemId', tag' > \in Q$. The second condition is trivial to achieve since A knows this knowledge.

Thus,

$$\Pr[Mint - Impersonate_{A,\Pi}(n) = 1 | r' = r] \cdot \Pr[r' = r] = \Pr[r' = r]$$

However, by the randomness of r,

$$\Pr[r' = r] = 2^{-n} \leq negl(n)$$

Case 2: $r' \neq r$

To evaluate $\Pr[Mint - Impersonate_{A,\Pi}(n) = 1 | r' \neq r]$, it suffices to reduce the evaluation to simply the probability of $Mac - forge_{A',\Pi'} = 1$. Logically, since $r' \neq r$ and if $itemId'$ has been seen before in one of the i^{th} queries from A, then it must be that A' outputs $itemId' \oplus r' \notin Q'$. Further, if $Mac_k(itemId' \oplus r')$ produces a valid tag, then both experiments Π and Π' output 1. Else, both output 0.

Hence,

$$\Pr\left[Mac - forge_{A',\Pi'}(n) = 1\right] = \Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1 | \, r' \neq r\right]$$

Since Π' is a secure MAC scheme, then

$$\Pr\left[Mac - forge_{A',\Pi'}(n) = 1\right] \leq negl(n).$$

Thus,

$$\begin{aligned} &\Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1 | r' \neq r\right] \cdot \Pr[r' \neq r] \\ &\leq \Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1 | r' \neq r\right] \times 1 \\ &\leq \; negl(n) \end{aligned}$$

Having evaluated both cases, we can now determine the probability of success for A on Π.

$$\begin{aligned} &\Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1\right] \\ &= \Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1 | r' = r\right] \cdot \Pr[r' = r] \\ &\qquad + \Pr\left[Mint - Impersonate_{A,\Pi}(n) = 1 | \, r' \neq r\right] \cdot \Pr[r' \neq r] \\ &\leq \; negl_1(n) + negl_2(n) \\ &\leq negl(n) \end{aligned}$$

This completes the proof.

3 The Lottery Scheme

3.1 NFT Lotteries

On August 28th, an NFT-based project known as “Loot Project” was introduced to the public [10]. The project offered 8,000 NFTs known as loot bags, with each bag associated with a bunch of items of varying rarity. Anyone could simply claim these bags for free and only needed to pay the standard Ethereum gas fee associated with the minting of these bags. The process for minting a particular bag was as simple as inputting any user-chosen bag id into the system and paying the minting fee. If the bag had not been minted by someone else, the bag would be successfully minted and owned by the user who minted it. The owner can then check all the items in the minted bag, especially the rare items which are of high value. These items are hierarchically classed on the factors of value and rarity in the same manner as items in a game like Dungeons & Dragons, where an example of a rare item would be “Divine Robe of the Fox”.

The aforementioned project is akin to a simple number lottery system with predetermined winning numbers. Participants would then pick a random number and hope that it was a winning number. These lottery-based giveaways are widely used in generating hype and attracting users to new NFT based projects. With such a demand, it would be beneficial for such lottery-based systems to be designed securely. The

potentially high valuation of the prizes also serves as a further motivation in creating secure online lottery systems. Typically, the actual implementation of an offline lottery scheme is hidden from the public. While this may be attributed to security reasons, it also allows for the conducting authority to make unfair adjustments to the lottery at their own discretion. For example, according to Rule 8.3 of the Singapore Pools TOTO Game Rules, the company may adjust the prizes offered at any time [14]. Kuacharoen proposed an online lottery system that claims to prevent this issue but also acknowledges that it is not perfect as there is still an element of trust needed in its design [13]. Having a lottery system based on the blockchain circumvents this by allowing the public to always monitor the system and verify the correctness of it. Any discrepancies because of, but not limited to, duplicated awards, fraudulent numbers, and any unannounced modifications to any part of the lottery system, would be easily discovered. However, such transparent systems are vulnerable against malicious participants who will attempt to discover the winning numbers through any means necessary. If a participant can discover that a particular number has a slightly higher odds of winning than the other numbers, then he is considered to have an advantage and the lottery would be deemed unfair.

3.2 Lottery Scheme

We now define the scheme called $Lottery$. Taking inspiration from a simple real life lottery scheme where the winning numbers are predetermined, $Lottery$ consists of a string called $prize$ and each of its bits is analogous to a lottery number. If the i^{th} bit is a 1, then it is considered a winning bit and is analogous to a winning lottery number. Just like how any participant in a real lottery doesn't know which number is a winning number, we need to mask $prize$ to hide information about the winning bits. To do this, we only reveal a string called $hiddenPrize$, that is generated after masking $prize$, to the adversaries. An adversary's job is to guess the winning bits in $prize$ with only knowledge about $hiddenPrize$. With this, we now provide the full definition of our $Lottery$ scheme:

Definition 2.1 Lottery: *A $Lottery$ consists of 3 probabilistic polynomial-time algorithms (Gen, $GenPrize$, $HidePrize$) such that:*

- *The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a key k with $|k| \geq n$.*
- *The prize-generation algorithm $GenPrize$ that takes as input a winning probability w such that $0 \leq w \leq 1$ and outputs a string $prize$ such that the fraction of 1s in $prize$ is equal to w.*
- *The prize-hiding algorithm $HidePrize$ that takes as input a key k and a string $prize$ and outputs a hidden-prize string $hiddenPrize$.*

For any string $hiddenPrize$, security of the lottery scheme means that an adversary shouldn't be able to guess the value of any i^{th} bit of $prize$ better than the initial winning probability w set for the lottery. This is analogous to real-life lotteries where

participants try to guess which numbers are the winning numbers. Ensuring that any adversary does not have such an advantage makes the lottery *fair*.

We now present the formal definition on the security property of *fairness* for a lottery scheme. Let $\Pi = (Gen, GenPrize, HidePrize)$, and consider the following experiment for an adversary A and parameters n, w:

The fair lottery game $Lottery - Fair_{A,\Pi}(n, w)$:

1. A key k is generated by running $Gen(1^n)$.
2. A string $prize$ is generated by running $GenPrize(w)$.
3. A string $hiddenPrize$ is generated by running $HidePrize(k, prize)$.
4. The adversary A is given $hiddenPrize$. The adversary eventually outputs i where i represents the i^{th} bit in $prize$ (indexed from 0).
5. A succeeds if and only if the guessed i^{th} bit in $prize$ is equal to 1. In event of success, the output of the experiment is defined to be 1.

A lottery is fair if no efficient adversary can succeed in the above experiment with non-negligible probability more than the wining probability w.

Definition 2.2: *A lottery is fair if, and only if, for any probabilistic polynomial time adversary, there exists a negligible function* $negl$ *such that:*

$$\Pr\left[Lottery - Fair_{A,\Pi}(n, w) = 1\right] \leq w + negl(n)$$

3.3 Constructing a Secure Lottery Scheme

At this juncture, a natural construction to use would be that of a one-time pad to mask $prize$. The indistinguishable property of a one-time pad encryption scheme implies that the ciphertext does not leak any information about individual bits of the plaintext. In our case, the $hiddenPrize$ is akin to a ciphertext and should not leak out any information about $prize$. Further, the function of a pseudorandom generator and the security it provides in a one-time pad encryption scheme would be useful in constructing our secure lottery scheme as well due to the similarities in both schemes.

Construction 2.3: Let G be a pseudorandom generator with expansion factor l. Define a $Lottery$ with security parameter n and a winning probability w as follows:

- Gen: On input 1^n, output a key $k \in \{0,1\}^n$
- $GenPrize$: On input w, output the string $prize \in \{0,1\}^{l(n)}$
- $HidePrize$: On input key $k \in \{0,1\}^n$ and $prize \in \{0,1\}^{l(n)}$, output the string $hiddenPrize = G(k) \oplus prize$.

Theorem 2.4: If G is a pseudorandom generator, then Construction 2.3 is *fair*.

Proof: Let Π denote the scheme in $Construction$ 1. We use a reduction approach to prove the fairness of Π; specifically, we construct a separate distinguishing game using an efficient distinguisher D that receives a string s and emulates the lottery game for an efficient adversary A. Let A be a probabilistic polynomial time adversary and let D be an efficient distinguisher. D emulates the lottery game for A in the following manner:

Distinguisher D:
D is given input 1^n, $s \in \{0,1\}^{l(n)}$ and string $prize \in \{0,1\}^{l(n)}$. D works as follows:

1. Create $hiddenPrize := s \oplus prize$.
2. Give $hiddenPrize$ to A and eventually receive i.
3. Output 1 if the i^{th} bit of $prize$ is equal to 1, else output 0.

D's ability to distinguish whether s was truly randomly generated or generated from a pseudorandom generator is directly related to A's ability to guess if the i^{th} bit of $prize$ equals to 1. If A can guess correctly with a non-negligible probability greater than the winning probability w, then it implies that A is somehow able to distinguish the origin of randomness for the string s. With this information, D should be able to likewise distinguish s with a non-negligible probability. This way, the security of the pseudorandom generator implies the fairness of Π.

Before analysing the probability of success of D in its distinguishing game, we construct a second game Π′ that is identical to Π except that $prize$ is XORed with a truly random string s of length $l(n)$ instead of the output from a pseudorandom generator to create $hiddenPrize$. In this game Π′, the probability of success for A is simply just the probability of winning w. To see why, consider the proof below:

i^{th} bit of $prize$	i^{th} bit of s	i^{th} bit of $hiddenPrize$ = i^{th} bit of $prize \oplus i^{th}$bit of s
0	0	0
1	0	1
0	1	1
1	1	0

Since s is randomly generated, then $\Pr[i^{th}\ bit\ of\ s = 0] = \Pr[i^{th} bit\ of\ s = 1] = 0.5$. For $prize$, $\Pr[i^{th} bit\ of\ prize = 1] = w$. Note that strings $prize$ and s are generated independently. Given the i^{th} bit of $hiddenPrize$ equals 1, the probability the i^{th} bit of $prize$ equals to 1 is:

$$\Pr[i^{th} bit\ of\ prize = 1 | i^{th} bit\ of\ hiddenPrize = 1]$$

$$= \frac{\Pr[\mathrm{i^{th}} bit\ of\ prize = 1 \cap i^{th} bit\ of\ hiddenPrize = 1]}{\Pr[i^{th} bit\ of\ hiddenPrize = 1]}$$

$$= \frac{w \times 0.5}{w \times 0.5 + (1-w) \times 0.5}$$
$$= w$$

A similar proof holds when the given i^{th} bit of $hiddenPrize$ equals to 0. In both cases, the probability of guessing whether the i^{th} bit of $prize$ equals to 1 is just w.

Now, if we observe the distinguishing game for D which runs A as a subroutine, we note that:

1. If the input s to D is a truly random string, then the probability of D outputting 1 in its own distinguishing game running A as a subroutine is equivalent to the probability of success of A in Π'. Thus, we have:

$$\Pr_{s \leftarrow \{0,1\}^{l(n)}}[D(s) = 1] = \Pr\left[Lottery - Fair_{A,\Pi'}(n, w) = 1\right] = w \quad (1)$$

2. Else if the string s provided to D is generated from a pseudorandom generator G on an input k, that is $s := G(k)$, then the probability of D outputting 1 in its own distinguishing game running A as a subroutine is equivalent to the probability of success of A in Π. Thus, we have:

$$\Pr_{k \leftarrow \{0,1\}^n}\left[D\left(G(k)\right) = 1\right] = \Pr\left[Lottery - Fair_{A,\Pi}(n, w) = 1\right] \quad (2)$$

Since G is a pseudorandom generator, then there must be a negligible function $negl$ such that:

$$\left| \Pr_{s \leftarrow \{0,1\}^{l(n)}}[D(s) = 1] - \Pr_{k \leftarrow \{0,1\}^n}\left[D\left(G(k)\right) = 1\right] \right| \leq negl(n)$$

Together with equations (1) and (2), we get:

$$|w - \Pr\left[Lottery - Fair_{A,\Pi}(n, w) = 1\right]| \leq negl(n)$$
$$=> \Pr\left[Lottery - Fair_{A,\Pi}(n, w) = 1\right] \leq w + negl(n)$$

This completes the proof.

Further discussion. We acknowledge that our proposed $Lottery$ scheme and its security definition of *fairness* are not applicable to all varying lottery systems in the world. Other conventional and popular lotteries include Toto and Lotto, which do not fit the mould of our defined $Lottery$ scheme. These separate lotteries will require their own lottery schemes and security definitions which we do not go into.

4 Conclusion

We have investigated two broad security implementations that underpin the implementations used in minting NFTs on third-party marketplaces: an impersonation

scheme and a lottery scheme. By devising the above schemes, constructions, and proofs, we demonstrated how analysts can qualify the security of such NFT-based implementations based on the qualities of whether an implemented scheme can be impersonated (*impersonate-able*), and whether the scheme implemented is *fair*.

NFTs have been hailed as a salient tool for wealth accumulation in the new age. It is a matter of time before the adoption of NFTs becomes mainstream. However, NFT implementations made by third-party marketplace service-providers in other areas beyond minting, such as transactions, fee deductions, and bidding, must continue to be hardened to minimize the risk of such attacks negatively impacting a burgeoning user base. This could result in devastating monetary losses to a significant portion of NFT-users. For NFTs to continue being reputable and embraced in accumulating wealth, the increasingly vital and relevant security qualities that make up NFTs must be thoroughly explored and refined.

References

1. Ante, Lennart: The non-fungible token (NFT) market and its relationship with Bitcoin and Ethereum (2021). https://ssrn.com/abstract=3861106.
2. Bonderud, D.: *Token Resistance: Tackling the New NFT Threat Landscape*. Security Intelligence (2021). https://securityintelligence.com/articles/new-threat-landscape-nfts/, last accessed 2 November 2021.
3. Crow, K., *Scammers and Hackers See New Frontier in NFT Art*. The Wall Street Journal (2021). https://www.wsj.com/articles/scammers-see-new-frontier-in-nft-art-11629896400, last accessed 2 November 2021.
4. Daniele, D.: *NFTs' Nifty Copyright Issues*. Mondaq (2021). https://www.mondaq.com/canada/copyright/1067734/nfts39-nifty-copyright-issues-, last accessed 30 October 2021.
5. Daubenschütz, T.: *What is sleepminting and will it ruin NFT provenance?* (2021). https://timdaub.github.io/2021/04/22/nft-sleepminting-beeple-provenance/, last accessed 3 November 2021.
6. Entriken W., Shirley, D., Evans, J., Sachs, N.: *EIP-721: Non-Fungible token standard. Ethereum Improvement Proposals (2018)*. https://eips.ethereum.org/EIPS/eip-721, last accessed 3 November 2021.
7. Fairfield, J.: *Tokenized: The law of non-fungible tokens and Unique Digital Property*. SSRN (2021). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3821102, last accessed 3 November 2021.
8. Finlow-Bates, K.: *How to sleepmint NFT tokens*. Medium (2021). https://kf106.medium.com/how-to-sleepmint-nft-tokens-bc347dc148f2, last accessed 3 November 2021.
9. Frankenfield, J.: Fungibility: *When interchangeability matters*. Investopedia (2021). https://www.investopedia.com/terms/f/fungibility.asp#:~:text=Fungibility%20is%20the%20ability%20of,equal%20value%20between%20the%20assets, last accessed 30 October 2021.
10. Hoffman, D.: Loot- randomized adventurer gear- no images or stats. intentionally omitted for others to interpret- no fee, just gas- 8000 bags totalopensea: Https://t.co/qsnrj1fd0netherscan: Https://t.co/bf9p0rshx2available via contract only. not audited. mint at your own risk pic.twitter.com/ulukzfayuk. Twitter (2021).

https://twitter.com/dhof/status/1431316631934967815?s=20, last accessed 2 November 2021.
11. Howcroft, E.: *NFT sales surge to $10.7 bln in Q3 as Crypto Asset Frenzy hits New highs*. Reuters (2021). https://www.reuters.com/technology/nft-sales-surge-107-bln-q3-crypto-asset-frenzy-hits-new-highs-2021-10-04/, last accessed 30 October 2021
12. Mashayekhi, R.: *Where the smart money in crypto investing is going next*. Fortune (2021). https://fortune.com/2021/07/29/crypto-investing-where-smart-money-going-next-andreessen-horowitz-nfts-cathie-wood-ark/, last accessed 2 November 2021.
13. Kuacharoen P.: *Design and Implementation of a Secure Online Lottery System*. In: Papasratorn B., Charoenkitkarn N., Lavangnananda K., Chutimaskul W., Vanijja V. (eds) Advances in Information Technology. IAIT 2012. Communications in Computer and Information Science, vol 344. Springer, Berlin, Heidelberg (2012).
14. Toto Game Rules (general). *TOTO Game Rules (General) | Singapore Pools.* (2021). https://www.singaporepools.com.sg/en/rules/pages/toto-game-rules-general.html, last accessed 2 November 2021.

AnonDB — Are Anonymous Databases Achievable?

Lyu Jiawen[1], Nguyen Khanh Duy[2], Ong Si Ying[3] and Toh Hong Xian[4]
[1] National University of Singapore, Singapore
e0376928@u.nus.edu
[2] National University of Singapore, Singapore
e0407663@u.nus.edu
[3] National University of Singapore, Singapore
e0202397@u.nus.edu
[4] National University of Singapore, Singapore
e0406975@u.nus.edu

Abstract. This paper introduces a generic healthcare database system representative of databases used in the healthcare industry today. With growing concerns over de-anonymization of databases such as the Netflix & IMDB incident, there is an increasing need to improve database anonymity. The concept of perfect anonymity is introduced: the best strategy for an attacker to identify one individual from the database should be a random guess, and knowledge of the records should not improve with repeated queries — this prevents the identification of individuals with multiple queries. The impracticality of a perfectly anonymous database is discussed, showing the tradeoff between anonymity and utility of database. Instead, the constraints of perfect anonymity are relaxed giving rise to a secured database. Secured databases with partial anonymity are enforced through security properties, such as *k*-security, *l*-security, and *t*-security. Constructions will also be introduced and evaluated using the aforementioned security properties, detailing the process of building a secured database from its raw un-anonymized form.

Keywords: Database, Security, Anonymity.

1 . Introduction

1.1 Terminologies

Definition 1.1.1. Sensitive Attributes. Defined as the most valuable information stored in the database. In the context of a healthcare database system, sensitive attributes refer to health information, such as an individual's disease.

Definition 1.1.2. Non-Sensitive Attributes. Defined as the remaining attributes that are not Sensitive Attributes (*Definition 1.1.1)*. In the context of a healthcare database system, non-sensitive attributes refer to information not related to their health, such as name, age, and gender.

Definition 1.1.3. Unique Identifiers. Defined as a subset of non-sensitive attributes (*Definition 1.1.2)* that can be used to uniquely identify a single record in a database (e.g., name, identification number).

Definition 1.1.4. Quasi-Identifiers. Quasi-Identifiers are subsets of Non-Sensitive Attributes (*Definition 1.1.2*) that are not themselves unique identifiers but can be combined with other quasi-identifiers to become unique identifiers.

Definition 1.1.5. Group. A group is defined as a collection of records with the same values for the quasi-identifiers.

1.2 Anonymity

Anonymity refers to a state of database where unique identifiers have been anonymized from databases such as names and identification numbers. This is usually done by data redaction, a process that sanitizes a database for privacy protection.

1.3 Perfect Anonymity

Intuition. For a database to be truly anonymous, the best possible strategy of an adversary Λ to identify an individual should be a random guess. In addition, the adversary's knowledge of the records must not improve with multiple database queries.

Definition 1.3.1. Perfect Anonymity. Let ID_{record} be the act of identifying a *record* from the database. Let *result* be the result from a database query. For all *result*, the database is perfectly anonymous if and only if, for a database of size *S*,

$$\forall\, record \in database,$$
$$Pr(ID_{record}) = \frac{1}{S}$$

$$\forall\, result \in possible\ query\ results,$$
$$Pr(ID_r | result) = Pr(ID_r) = \frac{1}{S}$$

Theorem 1.3.2. If the database is perfectly anonymous, for any adversary Λ who is able to query the database repeatedly, Λ will never identify any record *r*. The proof is trivial as it follows from *Definition 1.3.1*.

Construction 1.3.3. Perfectly Anonymous Database. To achieve a perfectly anonymous database, the database must not contain any information that distinguishes or helps to distinguish an individual from another. A possible construct is a database with all entries having identical values. Even if an attacker queries the database, the attacker will never know which exact record corresponds to an individual.

Feasibility of a Perfectly Anonymous Database. A perfectly anonymous database will have little to no use; they have constraints which severely limits the amount of information it can store and retrieve in preserving perfect anonymity. This presents a tradeoff between anonymity and utility in a database.

1.4 Security

Relaxing constraints of Perfect Anonymity. For a database to maintain a reasonable amount of utility, constraints of perfect anonymity from *Definition 1.3.1.* must be relaxed such that the probability of identifying an individual would be larger than $\frac{1}{S}$ but less than 1. The knowledge of the records could improve with each additional query, but it should never identify an individual record. These relaxed constraints are defined formally in the following sub-section.

Definition 1.4.1. Security. For all *q*, the database is secure if and only if*:*

$$Pr(l_p) < 1$$
$$Pr(l_n|q) = Pr(l_n) < 1$$

Databases with increasing security will be discussed in *Section 3* below.

2 . Proposed Cryptographic System

2.1 Healthcare Database

Table 2.1.1. Example instance of a healthcare database

	Non-sensitive Attributes				Sensitive
	Name	Postal	Age	Gender	Disease
1	Dennis	570321	18	Male	Cardio
2	Justin	576548	19	Male	Skin
3	Bob	570322	17	Male	None
4	Alex	576547	18	Male	Liver
5	John	641042	21	Male	Liver
6	Benedict	649430	28	Male	Cancer
7	Keith	641044	22	Male	Liver
8	Benedict	649431	27	Male	Cancer
9	Sally	501694	31	Female	Cancer
10	Jessica	509876	32	Female	None
11	Joy	501695	32	Female	Covid
12	Zoe	509877	35	Female	Diabetes
13	Dominic	133065	41	Male	Diabetes
14	Sam	136054	42	Male	Diabetes
15	Zeith	133066	51	Male	Diabetes
16	Dexter	136055	52	Male	Diabetes

Non-sensitive Attributes				Sensitive
Name	Postal	Age	Gender	Disease
*	57****	< 20	Male	Cardio
*	57****	< 20	Male	Skin
*	57****	< 20	Male	None
*	57****	< 20	Male	Liver
*	64****	20 - 40	Male	Liver
*	64****	20 - 40	Male	Cancer
*	64****	20 - 40	Male	Liver
*	64****	20 - 40	Male	Cancer
*	50****	20 - 40	Female	Cancer
*	50****	20 - 40	Female	None
*	50****	20 - 40	Female	Covid
*	50****	20 - 40	Female	Diabetes
*	13****	> 40	Male	Diabetes
*	13****	> 40	Male	Diabetes
*	13****	> 40	Male	Diabetes
*	13****	> 40	Male	Diabetes

The proposed cryptographic system ∏ is a healthcare database. In this discussion, the encryption scheme of the database is irrelevant given the assumption that the adversary does not have direct access to the system. The database can be queried by inputting the values of quasi-identifiers.

Size of database *S*. The healthcare database consists of a table with *S* number of records. That is, it contains the information of *S* individuals.

Generator *Gen*. The security parameter *n* is the number of bits of the database size *S* in binary. *Gen* produces *n* number of bits randomly and uniformly that forms *S* in binary and outputs *S* as an integer:

$$S = Gen(1^n)$$

Random Number Generator *Rand*. *Rand* is a cryptographically secure random number generator that generates a value from *lower* to *higher* inclusive randomly, uniformly and securely. *Rand* is used to generate a constant for security properties defined in *Section 3*.

$$Rand(lower, higher) \rightarrow constant$$

Query *Q* on System ∏. The query *Q* receives one input:

$$Q_{\prod}: input \rightarrow result$$
$$input = \{x: x \subseteq \text{quasi} - \text{identifiers}\}$$
$$result = \{x: x \subseteq \text{database records satisfying } input\}$$

System Assumptions. We assume that there is some challenger *C* who has access to ∏ and able invoke *Q* repeatedly. On the other hand, an adversary Λ does not have access to ∏; The adversary probes the system through an oracle O. Furthermore, Λ knows the attributes of the database system, such as age, gender, disease etc.

3 . Security Properties

3.1 *k*-security

Definition 3.1.1. *k*-anonymity. The database system ∏ is *k*-anonymous if every record in the table is indistinguishable from at least *k*-1 other records with respect to a set of quasi-identifiers [1].

Choice of *k*. The choice of *k* involves the tradeoff between anonymity and utility. The utility in this context is equivalent to the computational efficiency of identifying a sensitive value. The smaller *k* is, there would be lesser anonymity with higher utility, and the larger *k* is, there would be greater anonymity with lower utility.

It is to note that $k = 1$ and $k = S$, where *S* is the total number of records in the database system ∏, are meaningless — $k = 1$ will behave as an un-anonymized table, and $k = S$ will always return an entirely redacted table. Hence, in the subsequent parts of *k*-security, we will generate $k = Rand(2, S - 1)$.

Adversarial game for *k*-security. The goal of the adversary is to submit values of a set of quasi-identifiers $A_{\prod}$ that will result in bit $b' = 1$.

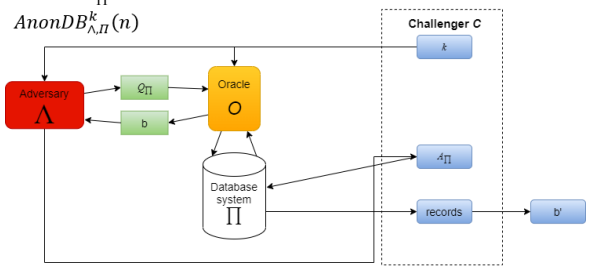


Fig. 3.1.1. *k*-security Adversarial Game

1. Challenger *C* will initialize the database system ∏ with size $S = Gen(1^n)$. *k* will be chosen as follows: $k = Rand(2, S\text{-}1)$. *k* is made known to the oracle *O* and adversary Λ.
2. Λ can query the *O* once with values of a set of quasi-identifiers $Q_{\prod}$.
3. *O* will query ∏ with $Q_{\prod}$ and count the number of records, *r*. *O* will return bit $b = 0$ to Λ if $r \geq k$. Otherwise, *O* will return bit $b = 1$ if $r < k$.
4. Given bit *b* value, Λ must submit values of a set of quasi-identifiers $A_{\prod}$ to *C*. *C* will query ∏ with $A_{\prod}$ and count the number of records, *r'*, returned by ∏.
5. If $r' < k$, Λ outputs bit $b' = 1$. Otherwise, Λ outputs bit $b' = 0$.

Theorem 3.1.2. *k*-security. The adversary Λ succeeds if $A_{\prod}$ gives bit $b' = 1$. The database system ∏ is *k*-secure if and only if, for any adversary Λ,

$$\Pr\left(AnonDB^{k}_{\Lambda,\Pi}(n) = 1\right) \leq negl(n)$$

Adversarial game for Multi *k*-security. The goal of the adversary is to submit values of a set of quasi-identifiers $A_{\prod}$ that will result in bit $b' = 1$.

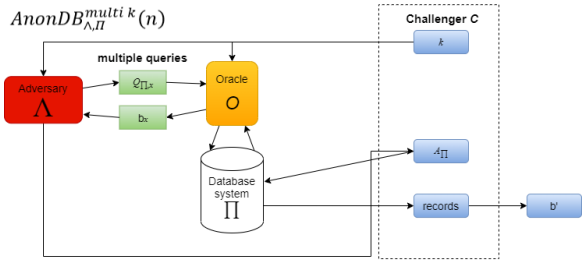


Fig. 3.1.2. Multi *k*-security Adversarial Game

1. Challenger *C* will initialize the database system ∏ with size S = Gen(1^n) *k* will be chosen as follows: *k* = *Rand(2, S-1)*. *k* is made known to the oracle *O* and adversary Λ.
2. Λ can **query the *O* as many times** as Λ wants with values of sets of quasi-identifiers $Q_{\prod,x}$, where *x* is the query number.
3. For each query $Q_{\prod,x}$, *O* will query ∏ with $Q_{\prod,x}$ and count the number of records, r_x. *O* will return bit $b_x = 0$ to Λ if $r_x \geq k$. Otherwise, *O* will return bit $b_x = 1$ if $r_x < k$.
4. Given the set of bit values b_x, Λ must submit values of a set of quasi-identifiers $A_{\prod}$ to *C*. *C* will query ∏ with $A_{\prod}$ and count the number of records, *r'*, returned by ∏.
5. If $r' < k$, Λ outputs bit $b' = 1$. Otherwise, Λ outputs bit $b' = 0$.

Theorem 3.1.3. Multi *k*-security. The adversary Λ succeeds if $A_{\prod}$ gives bit $b' = 1$. The database system ∏ is Multi *k*-secure if and only if, for any adversary Λ,

$$\Pr\left(AnonDB^{multi\ k}_{\Lambda,\Pi}(n) = 1\right) \leq negl(n)$$

What happens when the adversary has the ability to make multiple queries?
k-security is only guaranteed in any single-query system. Given that the database is not *k*-anonymous, once the adversary is able to make multiple queries, the adversary can observe multiple bit results returned by the oracle in different queries, and manipulate the subsequent queries to reduce the number of corresponding records, and ultimately submit a specially chosen $A_{\prod}$ which will result in:

$$\Pr\left(AnonDB^{multi\ k}_{\Lambda,\Pi}(n) = 1\right) = 1 > negl(n)$$

However, if we constrain the type of database system ∏ to be a *k*-anonymous table, it will be Multi *k*-secure, which we will discuss in *Theorem 3.1.6.*.

Theorem 3.1.4. If the system is Multi *k*-secure, it is also *k*-secure.
Proof by contrapositive. We will proceed to prove ¬(*k*-secure) → ¬(Multi *k*-secure).

Suppose ∏ is not *k*-secure. There exists an adversary Λ who can win the *k-secure Adversarial Game* with a non-negligible function ε according to *Theorem 3.1.2*. Λ

will query once, $Q_{\prod}$, and output $A_{\prod}$ which will produce bit $b' = 1$. Therefore, this means that the probability of Λ winning is: $\Pr\left(AnonDB_{\Lambda,\Pi}^{k}(n) = 1\right) > \varepsilon(n)$

With Λ as a subroutine in the *Multi k-secure Adversarial Game*, we construct another adversary Λ' that can query the oracle multiple times. Regardless of whether Λ' queries ∏, Λ' will output $A_{\prod}' = A_{\prod}$ which will result in bit $b' = 1$, which gives:

$$\Pr\left(AnonDB_{\Lambda',\Pi}^{multi\ k}(n) = 1\right) > \varepsilon(n)$$

By contrapositive, since ¬(*k*-secure) → ¬(Multi *k*-secure), this gives Multi *k*-secure → k-secure. Hence, if ∏ is Multi *k*-secure, it is *k*-secure too.

Intuition. If a system is Multi *k*-secure, it means that the system can prevent an adversary Λ with greater power from breaking the system. Hence, in the case where Λ cannot win the *Multi k-security Adversarial Game* with multiple queries to the ∏, Λ cannot possibly win the *k-security Adversarial Game* with a single query.

Construction 3.1.5. *k*-security. Given any database system ∏ and a fixed value of *k*, we have:
Vrfy: On input of values of a set of quasi-identifiers $Q_{\prod}$, output records if and only if $r \geq k$, where *r* is the number of records corresponding to $Q_{\prod}$.

Theorem 3.1.6. If the system is *k*-anonymous, it is also Multi *k*-secure.
Proof. Since ∏ will always return at least *k* records given the *Vrfy* function, oracle *O* will always output bit $b = 0$. Hence, even in a *Multi k-secure Adversarial Game*, the adversary Λ cannot attempt to learn any information about any individual or a collection of individuals by querying the *O*. Hence, it will always be Multi *k*-secure according to *Theorem 3.1.3*.

Attacks on *k*-anonymity. Even when working with a *k*-anonymous database, there are two main attacks that defeats *k*-anonymity.

Homogeneity attack. If the sensitive values within the group are homogeneous, it leaks the sensitive values of the individuals within the group to the adversary regardless the number of records in the group [1].

Background knowledge attack. In reality, attackers have access to other information of the victim(s). For example, given that the attacker know that the victim is a 78 years-old female and stays in Pasir Ris, if there are records with similar postal code and gender within the same group *O(<Janice, 51****, *8, Female, Skin>, <Rachel, 51****, *8, Dementia>)*, the attacker can guess with a higher probability that the victim is suffering from dementia given the victim's old age [1].

3.2 *l*-security

To prevent the k-anonymity attacks, a new property that is built on top of k-anonymity is introduced – *l*-diversity.

Definition 3.2.1. *l*-diversity. The database system ∏ is *l*-diverse if, for each group of records, there are at least *l* distinct values of sensitive values [1].

Choice of *l*. The choice of l depends on the choice of k in the previous section. From the *Definition 2.3.1.*, k refers to the size of the smallest group. The value of l considers the number of unique sensitive values within each group, thus $l \leq k$. Additionally, it must be the case that $l \geq 2$, else all records within the same group would have the same sensitive value.

Adversarial game for *l*-security. The goal of the adversary is to submit a set of quasi-identifiers $A_{\prod}$ that will result in bit $b' = 1$.

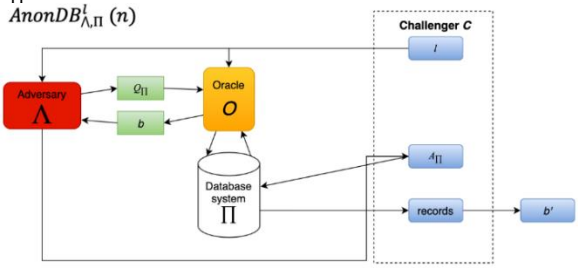


Fig. 3.2.1. *l*-security Adversarial Game

1. Challenger C will initialize the database system $\prod$ with size S = $Gen(1^n)$. Let k be the size of the smallest group ($2 \leq k \leq S - 1$). l will be chosen as follows: $l = Rand(2, k)$. k, l is made known to the oracle O and adversary Λ.
2. Λ can query the O once with a set of quasi-identifiers $Q_{\prod}$.
3. O will query $\prod$ with $Q_{\prod}$, count the number of records r and the number of distinct sensitive values v. O will return bit $b = 0$ to Λ if $r \geq k$ and $v \geq l$. Otherwise, O will return bit $b = 1$ if $r < k$ or $v < l$.
4. Given bit b value, Λ must submit values of a set of quasi-identifiers $A_{\prod}$ to C. C will query $\prod$ with $A_{\prod}$, count the number of records r' and the number of distinct sensitive values v', returned by $\prod$.
5. If $r' < k$ or $v' < l$, Λ outputs bit $b' = 1$. Otherwise, Λ outputs bit $b' = 0$.

Theorem 3.2.2. *l*-security. The adversary Λ succeeds if $A_{\prod}$ gives bit $b' = 1$. The database system $\prod$ is l-secure if and only if, for any adversary Λ,

$$\Pr\left(AnonDB^{l}_{\Lambda,\Pi}(n) = 1\right) \leq negl(n)$$

Adversarial game for Multi *l*-security. The goal of the adversary is to submit a set of quasi-identifiers $A_{\prod}$ that will result in bit $b' = 1$.

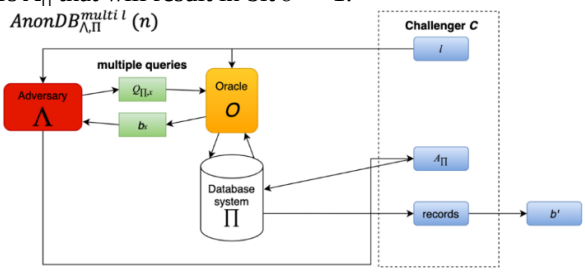


Fig. 3.2.2. Multi *l*-security Adversarial Game

1. Challenger *C* will initialize the database system ∏ with size $S = Gen(1^n)$. Let *k* be the size of the smallest group ($2 \leq k \leq S - 1$). *l* will be chosen as follows: $l = Rand(2, k)$. *k, l* is made known to the oracle *O* and adversary Λ.
2. Λ can **query the *O* as many times** as Λ wants with a set of quasi-identifiers $Q_{\prod,x}$, where x is the query number.
3. For each query $Q_{\prod,x}$, *O* will query ∏ with $Q_{\prod,x}$, count the number of records r_x and the number of distinct sensitive values v_x. *O* will return bit $b_x = 0$ to Λ if $r_x \geq k$ and $v_x \geq l$. Otherwise, *O* will return bit $b_x = 1$ if $r_x < k$ or $v_x < l$.
4. Given the set of bit values b_x, Λ must submit values of a set of quasi-identifiers $A_\prod$ to *C*. *C* will query ∏ with $A_\prod$, count the number of records *r'* and the number of distinct values *v'*, returned by ∏.
5. If $r' < k$ or $v' < l$, Λ outputs bit $b' = 1$. Otherwise, Λ outputs bit $b' = 0$.

Theorem 3.2.3. Multi *l*-security. The adversary Λ succeeds if $A_\prod$ gives bit $b' = 1$. The database system ∏ is Multi *k*-secure if and only if, for any adversary Λ,

$$\Pr\left(AnonDB^{multi\ l}_{\Lambda,\Pi}(n) = 1\right) \leq negl(n)$$

Theorem 3.2.4. Multi *l*-security implies *l*-security. Similar to *Theorem 3.1.4*, if a database system is Multi *l*-secure, it is also *l*-secure.

Construction 3.2.5. *l*-security. Given any database system ∏ and a fixed value of *l*. Let *k* be the size of the smallest group. We have:

Vrfy: On input of a set of quasi-identifiers $Q_\prod$, output records if and only if $r \geq k$ and $v \geq l$, where *r* is the number of records and *v* is number of distinct sensitive values corresponding to $Q_\prod$.

Theorem 3.2.6. If the system is *l*-diverse, it is also Multi *l*-secure.
Proof. Since ∏ will always return at least *k* records with at least *l* number of distinct sensitive values given the *Vrfy* function, oracle *O* will always output bit $b = 0$. Hence, even in a Multi *l*-secure adversarial game, the adversary Λ cannot attempt to learn any information about any individual or a collection of individuals by querying the *O*. Hence, it will always be Multi *l*-secure according to *Theorem 3.2.4*.

Theorem 3.2.7. *l*-security implies *k*-security. If the database system is *l*-secure, it is also *k*-secure where *k* is the size of the smallest group in the database system.
Proof by contrapositive. We will proceed to prove ¬(*k*-secure) → ¬(*l*-secure).

Suppose ∏ is not *k*-secure. There exists an adversary Λ who can win the *k-security Adversarial Game* with a non-negligible function ε according to *Theorem 3.1.2*. According to the *k-security Adversarial Game*, Λ will output $A_\prod$ whereby the resulting table has less than *k* rows.

With Λ as a subroutine in the *l-secure Adversarial Game*, we construct another adversary Λ'. Regardless of whether Λ' queries ∏, Λ' will output $A_\prod' = A_\prod$. The resulting

table will have less than k rows with the input A_{Π}', which will result in bit b' = 1, which gives:

$$\Pr\left(AnonDB^{l}_{\Lambda\prime,\Pi}(n) = 1\right) > \varepsilon(n)$$

By contrapositive, since ¬(k-secure) → ¬(l-secure), this gives l-secure → k-secure.

Theorem 3.2.8. Multi *l*-security implies Multi *k*-security. An extension to *Theorem 3.2.7.* to Multi variants still holds regardless of the number of queries that the adversary can make.

Attacks on *l*-diversity. Although l-diversity is built on top of k-anonymity to prevent Homogeneity and Background knowledge attacks, we still see a malicious attack on a l-diverse table – if the distribution of sensitive attributes of all groups are not the same, there will be some leakage of information for certain groups.

Skewness Attack. l-diversity does not consider overall distribution of sensitive attributes [3].

As seen in *Table 2.1.1.*, the probability that the sensitive attribute is *Cancer* in this table is 18.75%. However, the probability that the victim's disease is cancer given that victim is in group two is 50%. Therefore, if the adversary knows that the victim belongs to the second group, the adversary can improve his knowledge — that the probability that the victim has *Cancer* increases from 18.75% to 50%.

3.3 *t*-security

To prevent the possible skewness attack on a l-diverse system, a system with t-closeness is introduced, which considers the distributions of sensitive attributes in the system.

Definition 3.3.1. *t*-closeness. A system is said to have t-closeness if all groups achieved t-closeness. A group in a database table is t-closed if the earth mover's distance (EMD, the definition can be found in *Appendix 1*) between the distribution of the sensitive values in this group and the distribution of the sensitive values in the entire table is no more than a threshold t [2].

Choice of *t*. The smaller the value of t, the more similar the distribution of the sensitive values in a particular group is to the entire system. Since the denominator is always greater than or equal to the numerator in calculation of EMD, t can be any positive value in the range [0, 1]. If t equals to 0, the distribution of sensitive values in all groups is equal to the distribution of sensitive values in the entire system. If t equals to 1, the distribution of sensitive values in every group would be completely different from the distribution of the sensitive values in the entire system.

Adversarial game for *t*-security. The goal of the adversary is to submit a set of values of quasi-identifiers A_{Π} that will result in bit b' = 1.

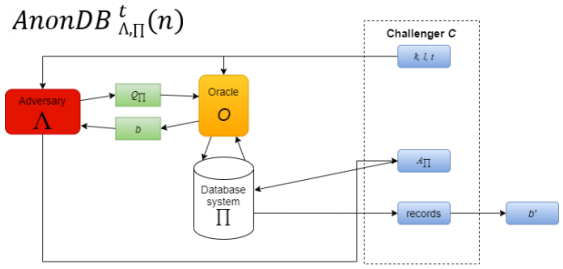


Fig 3.3.1. *t*-security Adversarial Game

1. Challenger *C* will initialize the database system ∏ with size S = *Gen*(1^n). Let k be the size of the smallest group ($2 \le k \le S$ - 1), *l* be the smallest number of distinct sensitive values among all groups ($2 \le l \le k$). *t* will be chosen as follows: *t* = *Rand(0, 1)*. *k*, *l*, *t* is made known to the oracle *O* and adversary Λ.
2. Λ can query the *O* once with a set of quasi-identifiers $Q_{\prod}$.
3. *O* will query ∏ with $Q_{\prod}$ and then calculate the EMD, *d*, between records and the entire system, and count the number of records *r* and the number of distinct sensitive values *v*. *O* will return bit *b* = 0 to Λ if $r \ge k$, and $v \ge l$ and $t \ge d$. Otherwise, *O* will return bit *b* = 1.
4. Given bit *b* value, Λ must submit values of a set of quasi-identifiers $A_{\prod}$ to *C*. *C* will query ∏ with $A_{\prod}$ and calculate the EMD, *d'*, between records and the entire system, and count the number of records *r'* and the number of distinct sensitive values *v'*.
5. Λ outputs bit *b'* = 1 if $k \ge r'$, or $l \ge v'$ or $d' \ge t$. Otherwise, Λ outputs *b'* = 0.

Theorem 3.3.2. *t*-security. The adversary Λ succeeds if single $Q_{\prod}$ gives $d > t$. The database table is *t*-secure if, for any adversary Λ,

$$\Pr\left(AnonDB^{\ t}_{\Lambda,\prod}(n) = 1\right) \le negl(n)$$

Adversarial game for Multi *t*-security. The goal of the adversary is to submit a set of values of quasi-identifiers $A_{\prod}$ that will result in bit *b'* = 1.

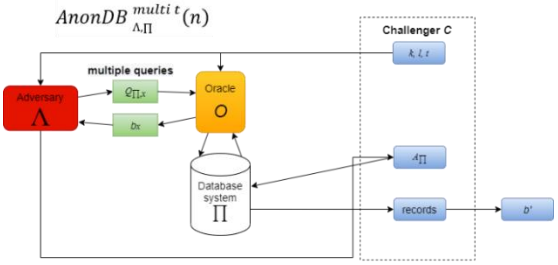


Fig 3.3.2. Multi *t*-security Adversarial Game

1. Challenger *C* will initialize the database system ∏ with size S = *Gen*(1^n). Let k be the size of the smallest group ($2 \le k \le S$ - 1), *l* be the smallest number of

distinct sensitive values among all groups ($2 \leq l \leq k$). t will be chosen as follows: $t = Rand(0, 1)$. k, l, t is made known to the oracle O and adversary Λ.

2. Λ can **query the O as many times** as Λ wants with sets of quasi-identifiers $Q_{\prod,x}$, where x is the query number.
3. For each query $Q_{\prod,x}$, O will query $\prod$ with $Q_{\prod,x}$ and calculate the EMD, d_x, between the records and the entire system, and count the number of records r_x and the number of distinct sensitive values v_x. O will return bit $b = 0$ to Λ if $r_x \geq k$, and $v_x \geq l$ and $t \geq d_x$. Otherwise, O will return bit $b = 1$.
4. Given the set of bit values b_x, Λ must submit values of a set of quasi-identifiers $A_{\prod}$ to C. C will query $\prod$ with $A_{\prod}$ and calculate the EMD, d', between the records returned by $\prod$ and the entire system, and count the number of records r' and the number of distinct sensitive values v'.
5. Λ outputs bit $b' = 1$ if $k \geq r'$ or $l \geq v'$ or $d' \geq t$. Otherwise, Λ outputs $b' = 0$.

Theorem 3.3.3. Multi *t*-security. The adversary Λ succeeds if single $Q_{\prod}$ gives $d > t$. The database system is Multi t-secure if, for any adversary Λ,

$$\Pr\left(AnonDB_{\Lambda,\prod}^{multi\ t}(n) = 1\right) \leq negl(n)$$

Theorem 3.3.4. Multi *t*-security implies *t*-security. Similar to *Theorem 3.1.4*, if a database system is Multi t-secure, it is also t-secure.

Construction 3.3.5. *t*-security. Given any database system $\prod$ and a fixed value of t, let k be the size of the smallest group, l be the smallest number of distinct sensitive values among all groups. We have:

Vrfy: On input of a set of quasi-identifiers $Q_{\prod}$, output records if and only if $r \geq k$ and $v \geq l$ and $d \leq t$, where r is the number of records, v is the number of distinct sensitive values, d is the EMD distance between the result group and the whole system $\prod$.

Theorem 3.3.6. If the system is t-closed, it is also Multi t-secure.

Proof. Since $\prod$ will always return at least k records with at least l number of distinct sensitive values, and the EMD between the records and the entire system is no larger than t, and the given the *Vrfy* function, oracle O will always output bit $b = 0$. Hence, even in a Multi t-secure adversarial game, the adversary Λ cannot attempt to learn any information about any individual or a collection of individuals by querying the O. Hence, it will always be Multi t-secure according to *Theorem 3.3.3*.

Theorem 3.3.7. *t*-security implies *l*-security. If t-security is achieved in a system, with k as the smallest size of the groups, and l the smallest number of distinct sensitive values in a group, then the system also achieves l-security.

The proof is similar to *Theorem 3.2.7,* it can be proved by contradiction considering the adversarial game for t-security.

Theorem 3.3.8. Multi *t*-security implies Multi *l*-security. This theorem is the extension of *Theorem 3.3.7* to Multi variants, and it still holds regardless of the number of queries that the adversary can make.

Attacks on *t*-closeness. If a system is *t*-secure, it is still not guaranteed to be anonymous with differential attacks on the system.

Differential Attack. Adversaries can get the sensitive value of an individual record after learning some information about the sensitive values in the group and observing all other records within the group.

The specific details of such an attack and how this attack can be resolved is left as a possible extension to our project.

4 . Conclusion

4.1 *Summary of Security Properties Discussed*

Table 4.1.1. Security Properties of Databases

Properties			Defs	Constructions	Proofs
Perfect Anonymity			1.3.1	1.3.3	
↓					
Security			1.4.1		
k-security	←	multi *k*-security	3.1.2	3.1.5	Theorem 3.1.4
↑		↑	3.1.3		Theorem 3.1.6
l-security	←	multi *l*-security	3.2.2	3.2.5	Theorem 3.2.4
			3.2.3		Theorem 3.2.6
↑		↑			Theorem 3.2.7
					Theorem 3.2.8
t-security	←	multi *t*-security	3.3.2	3.3.5	Theorem 3.3.4
			3.3.3		Theorem 3.3.6
					Theorem 3.3.7
					Theorem 3.3.8

Are Anonymous Databases Achievable? The idea of an "anonymous database" seems to present a false dichotomy that a database is either good (anonymous) or bad (non-anonymous). However, it is important to consider that a fully anonymized database can never store useful information — our relentless pursuit for data anonymity will mean that any form of data storage will be useless.

Therefore, considering the trade-off between anonymity and utility, we believe that a secure database exists on the spectrum of anonymity. In this report, we have defined and proposed databases that are secure. Since we discussed three main ideas in the report — *k*, *l* and *t* security, we believe database managers can adapt these ideas and properties to ensure that their database systems are secured. A possible extension of this project could consider the circumvention of differential attacks on t-secure systems by employing epsilon-differential privacy on the system.

Appendix 1

Earth mover's distance (EMD). Let $P = \{(p_1, w_{p_1}), \ldots, (p_m, w_{p_m})\}$ be the first signature with m clusters, where p_i is the cluster representative and w_{pi} is the weight of cluster. In a database system, P is the group, and the cluster representative is the cluster with the same sensitive attributes, and weight is the probability of the sensitive attributes appearing in the cluster. $Q = \{(q_1, w_{q_1}), \ldots, (q_n, w_{q_n})\}$ be the second signature with n clusters. $D = [d_{i,j}]$ the ground distance matrix where $d_{i,j}$ is the ground distance between p_i and q_j. A flow $F = [f_{i,j}]$, with f_{ij} being the flow between p_i and q_j, minimizes the overall cost $WORK(P, Q, F) = \sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}$, subject to the following constraints (1)-(4):

(1). $f_{ij} \geq 0, \ 1 \leq i \leq m, 1 \leq j \leq n$

(2). $\sum_{j=1}^{n} f_{ij} \leq w_{pi}, \ 1 \leq i \leq m$

(3). $\sum_{i=1}^{m} f_{ij} \leq w_{qj}, \ 1 \leq j \leq n$

(4). $\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = \min\left(\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qj}\right)$

The Earth mover's distance (EMD) between P, Q is:

$$EMD(P, Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}\, d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij}}$$

References

1. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity, https://personal.utdallas.edu/~mxk055100/courses/privacy08f_files/ldiversity.pdf.
2. Ninghui, L., Tiancheng, L., Venkatasubramanian, S.: t-Closeness: Privacy Beyond k-Anonymity and ℓ-Diversity, https://www.cs.purdue.edu/homes/ninghui/papers/t_closeness_icde07.pdf.
3. Shmatikov, V.: CS380s - Theory and Practice of Secure Systems, https://www.cs.utexas.edu/~shmat/courses/cs380s_fall09/21kanon.ppt.

Security Formulation for Anonymity on Health Database

Lim Ethan (A0199556H) and Zhang Yilin (A0194570Y) and Nandar Soe (A0236589L)

Abstract. Anonymizing or removing identifiable patient information is the first important step in complying with regulations and addressing privacy concerns[2]. In this paper we will review existing anonymization techniques on health databases. We also provide an overview of possible anonymous data attacks. We will apply several attacks on k-anonymity databases to demonstrate that although k-anonymity is necessary, only k-anonymity is not sufficient to protect patient privacy. We will then discuss methods of L-diversity to protect against such attacks and succeed in protecting patient privacy.

Keywords: K-anonymity· L-diversity· Healthcare data

1 Introduction

Taking example of a healthcare system, the database is very likely to store many types of personal information on patients and their medical records. Hence, it is very likely that this set of information in the original database will also include personal identifiers such as NRIC number, phone number and birthday, which we would not want to have disclosed. With many differing attributes for each entry in a database, each record becomes very possibly uniquely identifiable, allowing a malicious user of the database to identify and extract personal information. Therefore, there is this need for anonymisation to separate the information from the individual.

There are regulations such as the GDPR[1] requiring data anonymity or the removal of personally identifiable or sensitive information before processing a knowledge extraction task or query. We will be using databases that do not contain personal identifiers such as names in our examples.

Although the data is anonymized, it can still be exposed to multiple attacks. A commonly cited example would be analysis of the 1990 US census by Latanya Sweeney[4]. Sweeney was a graduate student then who was able to use voter rolls, combined with the dataset released by the Massachusetts Group Insurance Commission (which was claimed to be anonymized to protect data), to identify the data belonging to the then governor of Massachusetts William Weld. Her work shows the vulnerability that still remains present even with anonymization in place, which is why we need to examine this concept of anonymization more seriously and carefully.

In our paper we will mainly be looking at the anonymization of a healthcare database, using techniques such as k-anonymity and L-diversity.

2 Related work

From this paper[6] it examines systems which employ the use of k-anonymized datasets. In their analysis, they highlight that k-anonymity on its own is insufficient, due to the problem that k-anonymity does not guarantee privacy when the attacker has background knowledge, and also that it cannot defend against the homogeneity attack. And therefore, pushing the need for L-diversity.

3 Basic definitions and techniques

Anonymisation is to convert personal data into “anonymised data” by applying a range of anonymization techniques. And the purpose of anonymisation is to reduce the original information in the dataset by some extent and reduce the “identifiability” of one or more individuals from the original dataset.

There also are many possible techniques of anonymisation that we may not use but will mention here: character masking, aggregation, and pseudonymisation. Character masking is to change some characters of a data (partially) into “*” where the part it hides can provide extension of anonymity. Aggregation is converting the accurate value in a dataset into summarised values. It is used when aggregated data is sufficient to store in the database and individual reports are not required. Pseudonymisation is to replace the identifying data with some made up and non-repeatable values. It is used when data is required to be uniquely distinguished.

3.1 Basic definitions

Identifiers are a set of attributes that can directly identify a person.

Quasi-identifiers are a set of attributes that can be associated with external information and can be used to re-identify or reduce the uncertainty of all or part of individual information.

Sensitive Information is a set of attributes that contain sensitive information of the user stored in the database. In the healthcare database, sensitive information can be disease, medication etc.

Linkage attack: Attributes including zip code, age or gender can be used by adversaries. If a certain quasi-identifier appears both in an external database (containing name of individual), and published microdata (containing sensitive information of individual), then the adversary can combine two databases to get both name and sensitive information[5].

3.2 Techniques for generating anonymous models

Generalization is to convert the accurate data into a range of less accurate data. It is mostly used when generalised values are still useful for the intended purpose. The k value in k-anonymity will affect the range chosen for generalization.

Suppression is to remove one or more attributes (entire columns) in a database. It is used when the data being supressed is not required for the intended purpose, or the attribute could not be appropriately anonymised[7].

Pseudonymisation is to replace the identifying data with some made up and non-repeatable values[6]. It is used when data is required to be uniquely distinguished. In the healthcare database, it can be used to hide the sensitive information. For example, the disease burn due to water-skis on fire can be replaced with a unique code V9107XD.

4 Overview of the specific formulations

4.1 Naïve k-anonymous

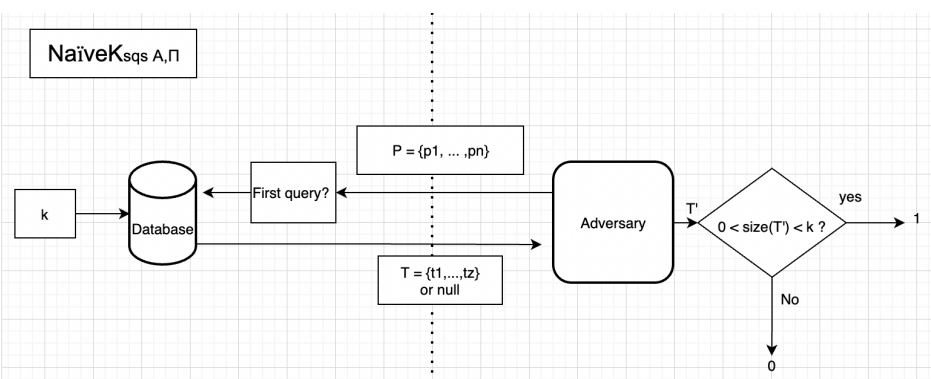
Starting off we will first look at a naïve first attempt to establish k-anonymity. That is, a system which operates by making sure that any query to the target database shall return at least k records in response, or it shall not respond at all. This naïve implementation that we will look at is of course not a practical one, and the purpose of our going through it is to show how careless implementation can sound correct to the regular person but actually be insecure.

Definition 1.(Quasi-identifier-no-restriction) *If a quasi-identifier is set to value *, then the quasi-identifier will not be used to filter any tuples and accept any value returned by the database.*

Definition 2. (Naïve-k-anonymity) *A given table T of data will be naïvely-k-anonymous if for every single possible value qi ∈ Qi in the set of Quasi-Identifiers, there exists at least k tuples t1,...,tk ∈ T that share the same qi.*

Game 1. (NaïveKsqs A,Π) *Let Qi be the set of values of the ith quasi-identifier. A challenger generates k>1, as input for a Naive-k-anonymous system with n quasi-identifiers. An adversary chooses any Quasi-Identifier values P = {p1, ... pn}, in which every pi ∈ Qi ∪ {'*'}, and sends it to the challenger. The challenger only accepts one query, and stops thereafter. Then, if the set of tuples T in the database matching P is greater than or equal k, they are given to the adversary. If not, nothing is given to the adversary. The adversary succeeds if he ends up with a set of tuples T' such that 0 < size(T') < k.*

Diagram 1. Naïve-k-anonymous game.



In this scenario, an adversary has the option to choose any combination of Quasi-Identifiers as a query to the database. However, regardless what query the adversary tries to make, the naive system will only ever allow a response that results in the adversary having a table T satisfactory to being either bigger than k or zero.

Here we will refer to this system as being Single-Query Secure, which will be explained why shortly below. An example of intersection attack will also show why the naïve system is not at all secure.

Definition 3. (Single-Query Secure) *A data anonymization system Π is considered to be Single-Query Secure if for any Adversary, NaïveKsqs A,Π = 0.*

Construction 1. Naïve-k-anonymous System
Given a database DB,

- Init: set system anonymity to integer k, apply suppression to remove all identifiers, and apply generalization techniques to each quasi-identifier in DB so that DB is naïvely-k-anonymous.
- Query: on input P = {p1, … pn}, find set T = {t1, … tz} in DB that matches with P.
- Response: Return T if size(T) >= k, else return null.

Theorem 1. *If k > 1, Construction 1 is Single-Query Secure.*

Proof: For any Query call, the naïvely-k-anonymous database DB has an appropriate response. For a Query that only looks up a single attribute: there are at least k tuples that are able to fill T, providing a valid response to the adversary. In addition, the adversary only receives information from table T and nowhere else, so his output T' should be exactly the same as T, resulting in $\text{NaïveKsqs}_{A,\Pi} = 0$. For a Query that looks up multiple attributes: if there are less than k tuples, null is returned in Response, leaving the adversary with T of size 0. Therefore size(T') = 0 and again $\text{NaïveKsqs}_{A,\Pi} = 0$.

Intersection attack: Though the Naïve-k-anonymous system is now secure for any query sent to it, security is lost once the adversary makes query number 2, 3,... and so on. We shall use the following 3-anonymised database in Table 1, and show that multiple queries to the database allow the adversary to obtain T' such that 0 < size(T') = 1 < k.

Table 1. Naïve-3-anonymous database.

Age	Sex	Zip Code	Disease
[40-50]	F	[10000-15000]	Z631
[40-50]	F	[10000-15000]	V9107XD
[40-50]	M	[15000-20000]	W52XXXD
[40-50]	F	[15000-20000]	Z631
[50-60]	M	[15000-20000]	W52XXXD
[50-60]	M	[10000-15000]	V9107XD
[50-60]	M	[10000-15000]	V9107XD

Table 2. Query 1: all Males.

M	W52XXXD
M	W52XXXD
M	V9107XD
M	V9107XD

Table 3. Query 2: all Females.

F	Z631
F	V9107XD
F	Z631

Table 4. Query 3: all ages 40-50.

[40-50]	Z631
[40-50]	V9107XD
[40-50]	W52XXXD
[40-50]	Z631

Table 5. Query 4: zip code 15000-20000.

[15000-20000]	W52XXXD
[15000-20000]	Z631
[15000-20000]	W52XXXD

So our adversary will make three separate queries to the database (Table 1). We can see that for each query, in Tables 1, 2 and 3, the number of tuples returned to the adversary is satisfactory to the requirement size(T') greater than or equal 3. So the naïve-3-anonymous system works correctly for now.

However, upon taking the three queries together, the adversary is suddenly able to obtain more information than he had before from each of them individually. Taking a look at Table 2, it can be seen that 2 Males have disease W52XXXD and 2 have disease V9107XD. From Table 3, no female has disease W52XXXD. From Table 4, it can be seen that one of those Males is of age range 40-50. From Table 5, it can be seen that both Males with disease W52XXXD stay in zip 15000-20000.

Intersecting this information, an adversary is able to reconstruct the following result T' in Table 6.

Table 6. Reconstructed record, size(T') = 1.

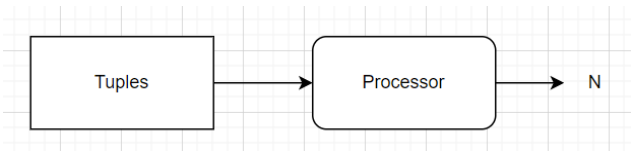
[40-50]	M	[15000-20000]	W52XXXD

4.2 Advanced k-anonymous

As Naïve-k-anonymity is no longer secure when querying multiple times, we need to apply more advanced anonymity techniques. Therefore, an advanced k anonymity scheme is introduced. A system of advanced k anonymity operates by making sure that if the adversary is allowed to query multiple times, then each time it will return no less than k records. Additionally, after the adversary query multiple times, the intersection of all the records after each query should be no less than k.

Definition 4. (Processor) *A processor will take in tuples returned by the database, and group the tuples with all the same quasi-Identifiers into one same group, called Ti. After the processor generates all the groups {T1, T2, …, Tn}, it will generate a number N, which is the smallest size of group among all the groups. This can be represented by Process(T) = N.*

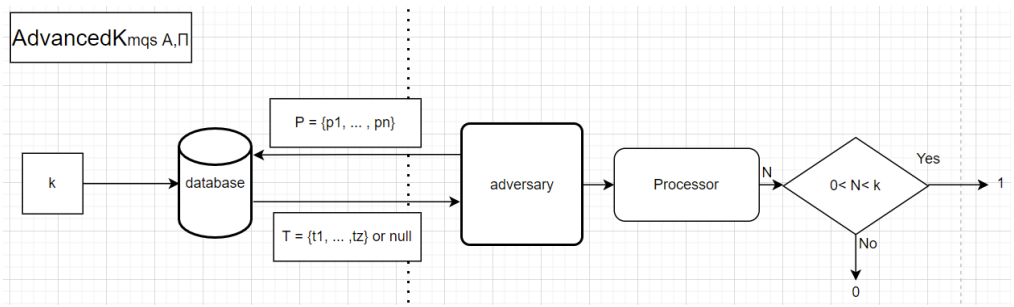
Diagram 2. Processor generates count of smallest size group = N.



Definition 5. (Advanced-k-anonymity) *Let Qi be the set of values of the ith quasi-identifier. A given table T of data with n quasi-Identifiers will be advanced-k-anonymous if for every possible value of pi ∈ Qi in the set of Quasi-Identifiers, the set of tuples that share the same P = {p1, … pn} returned is T and Process(T) is either ≥ k or 0.*

Game 2. (AdvancedKmqs A,Π) *Let Qi be the set of values of the ith quasi-identifier. A challenger generates k>1, as input for an Advanced-k-anonymous system with n quasi-Identifiers. An adversary chooses any of Quasi-Identifiers values P = {p1, ..., pn}, for every pi, pi ∈ Qi ∪ {"*"}, and sends it to the challenger. The adversary is allowed to query for multiple times to the challenger, returned with a set of tuples T. The adversary then gets N = Process(T). If 0 < N < k, then the adversary succeeds.*

Diagram 3. Advanced-k-anonymous game.



Definition 6. (Multi-Query secure) *A data anonymization system Π is considered to be Multi-Query Secure if for any Adversary, AdvancedKmqs A,Π = 0.*

Construction 2. Advanced-k-anonymous System
Given a database DB,

- Init: set system anonymity to integer k. Apply suppression to remove all identifiers, and apply generalization techniques to each quasi-identifier in DB so that DB is Advanced-k-anonymous.
- Query: on input P = {p1, … pn}, find set T = {t1, … tz} in DB that matches with P.
- Response: Return T if size(T) >= k, else return null.

Theorem 2. *If k > 1, Construction 2 is Multi-Query Secure.*

Proof: Since the database after init is Advanced-k-anonymous, there will always be at least k tuples that share all the same quasi-identifiers. Then when the adversary calculates Process(T), he will always be given the value greater than or equal to k. Therefore, *AdvancedKmqs A,Π will equal to 0 all the time.*

Theorem 3. *If k > 1, Construction 2 is secure under intersection attack.*

Proof: The Advanced-k-anonymous system is secure under intersection attack. Since the database after init is Advanced-k-anonymous, there will always be at least k tuples that share all the same quasi-identifiers. Therefore, no matter the attacker's queries, when he tries to do the intersection of all queries, it will only get an intersection of queries which has the size either equal to 0 or >= k. As k is greater than 1, he will always get the intersection of tuples with size greater than or equal to 2. Therefore, it is secure under multi-query intersect attack.

Theorem 4. *If a system is Multi-Query Secure, it is also Single-Query Secure.*

Proof: Here assume a system Π is Multi-Query Secure over for input k, i.e., Multi-Query-Secure(Π, k). Supposing that Π is somehow not Single-Query Secure, i.e., ¬ Single-Query-Secure(Π, k), it would mean that the adversary can obtain table T where 0 < size(T) < k from just a single query to the database. However, this would also result in the same undesired table T in the very first query of the multi-query scenario, and thus implies ¬ Multi-Query-Secure(Π, k). A contradiction.

However, even with being Multi-Query Secure, the system is still not secure under several attacks including homogeneity attack and background knowledge attack[3].

Table 7. Advanced-3-anonymous database.

Age	Gender	Disease
[0, 50]	M	Z631
[0, 50]	M	W52XXXD
[0, 50]	M	W52XXXD
Greater than 50	F	Z631
Greater than 50	F	Z631
Greater than 50	F	Z631
[0, 50]	F	V9107XD
[0, 50]	F	W52XXXD
[0, 50]	F	W52XXXD

Table 8. Query 1: all Females ages 0 to 50.

[0, 50]	F	V9107XD
[0, 50]	F	W52XXXD
[0, 50]	F	W52XXXD

Table 9. Query 2: all persons age greater than 50.

Greater than 50	F	Z631
Greater than 50	F	Z631
Greater than 50	F	Z631

Homogeneity attack: The adversary will make 1 query to the database table shown above. We can see from Query 2. The set of tuples returned is T_2 = {{"Greater than 50", "F", "Z631"}, {"Greater than 50", "F", "Z631"}, {"Greater than 50", "F", "Z631"}}, and although Process(T_2) = 3, since the value of disease is all the same, the adversary will know that a patient who is greater than 50 has disease with code Z631.

Background-knowledge attack: The adversary will make 2 queries to the database table shown above. From Query 1 we are returned a set of tuples returned is T_1 = { {"[0, 50]", "F", "V9107XD"}, {"[0, 50]", "F", "W52XXXD"},{ "[0, 50]", "F", "W52XXXD"}}, Process(T_1) = 3. However, there are only two different values of disease. If the attacker has the background knowledge that the female being attacked with age [0, 50] has much lower possibility of

having disease with code W52XXXD compared with V9107XD, then the adversary can even conclude that the female has disease with code V9107XD.

Since multi-query security is not secure against homogeneous attack and background attack, more techniques need to be applied to guarantee each block of tables have enough different diverse sensitive values.

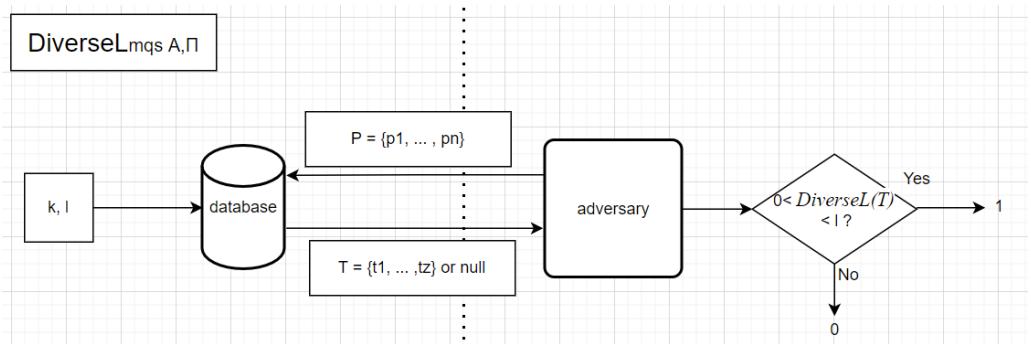
4.3 L-diverse + Advanced k-anonymous

Definition 7. (DiverseL(T)) *group the tuples with all the same quasi-Identifiers into one same group, called Ti, After the processor generates all the groups {T1, T2, …, Tn}, it will calculate the number of different sensitive information within each group, and output the smallest number.*

Definition 8. (Diverse-L-over-advanced-k-anonymous) *Let Qi be the set of values of the ith quasi-identifier. A given Advanced-k-anonymous table T of data with n quasi-Identifiers will be diverse-L-over-advanced-k-anonymous if for every possible value of pi ∈ Qi in the set of Quasi-Identifiers, the set of tuples that share the same P = {p1, … pn} returned is T. Where DiverseL(T) must either be ≥ l or 0.*

Game 3. (*DiverseLmqs A,Π*) *Let Qi be the set of values of the ith quasi-identifier. A challenger generates k>1 and L>1, as input for an Diverse-L-over-advanced-k system. An adversary chooses any Quasi-Identifiers values P = {p1, …, pn}, in which every pi ∈ Qi ∪ {"*"}, and sends it to the challenger. The adversary is allowed to query for multiple times to the challenger, the adversary will be returned a set of tuples T. If 0 < DiverseL(T) < l, then the adversary succeeds.*

Diagram 4. Diverse-L-over-advanced-k-anonymous game.



Definition 9. (L-diversity-over-multi-query secure) *A diverse-L-over-advanced-k system Π is considered to be L-diversity-over-multi-query Secure if for any Adversary, DiverseLmqs A,Π = 0.*

Construction 3. Diverse-L-over-advanced-k System
Given a database DB,

- Init: set system anonymity to integer k, set diversity to L. Apply suppression to remove all identifiers, and apply generalization techniques to each quasi-identifier in DB so that DB is Advanced-k-anonymous.
- Query: on input P = {p1, … pn}, find set T = {t1, … tm} in DB that matches with P.
- Response: Return T if size(T) >= k and DiverseL(T) >= L, else return null.

Theorem 5. *If k > 1 and L > 1, then Construction 3 is L-diversity-over-multi-query secure.*

Proof: The Construction will only respond if DiverseL(T) is greater than L for any query the adversary makes. Therefore, when the adversary calculates DiverseL(T), he or she cannot generate a value which is greater than 0 and smaller than L. Therefore, DiverseLsqs A,Π will always be 0 for any query. Therefore, Construction 3 is L-diversity-over-multi-query secure.

Theorem 6. *If k > 1 and L > 1, then Construction 3 is secure under Homogeneous attack.*

Proof: If the DB returns some tuples, then DiverseL(T) must be greater than or equal to L. Therefore, since L is greater than 1, DiverseL(T) must be greater than 1. Therefore, the situation of homogeneous attack, in which all the sensitive values output are the same, will not occur. Therefore the system is secure under homogeneous attack.

5 Discussion

We have finished off with a proof that shows that a L-diverse-over-advanced-k system that employs both techniques of advanced-k-anonymity and L-diversity is able to cover more ground and be more secure against attacks like the homogeneity attack, comparatively more so than our very first formulation of a Naïve-k-anonymous System. And also while the system is not completely secure against background knowledge attacks, it is at least more resilient: An adversary would still require multiple pieces of background knowledge to be able to rule out the L-1 other differing sensitive attributes within the table T.

These attacks and solutions that we have shown, gives a picture of some of the motivations and reasons behind the various techniques that have come to be used in the world of data anonymization. We first start off with the problem of needing to anonymize the data we publish. Then, we immediately see that a careless implementation of anonymization needs to be avoided, which led to the use of k-anonymity. And subsequently, there was the issue of homogeneity and background attacks that required the use of L-diversity. And it is here we see this pattern of problems, followed by solutions, followed by problems, … This pattern continues on, even with stronger notions of anonymity! - Take for example L-diversity, which itself has its own issues we have not discussed, that has to be later solved by yet another technique called T-closeness. Each solution in itself is a patch on top of a patch, dealing with problems that arise as they come in the whole anonymization process of the database.

So while we indeed have gone through some formulations making use of such ‘solutions’, our goal or so to speak of is not to encourage the use of a particular technique, but rather we would like to draw attention to frailty in which these techniques have been conceived. In the midst of searching for a method for anonymizing data, one should be aware of the possible imperfections that come along with each of these techniques. None of the above mentioned methods should be a be-all-end-all solution, since none of them are able to be a hard guarantee for privacy.

In addition, we also did not cover the aspect of usefulness of the anonymized data, which is its own separate issue by itself. By using some of the techniques like T-closeness for example, results from a search query end up with a similar distribution of sensitive attributes as the entire database population. This not only diminishes the value of having this database, but the T-close database still remains vulnerable to some adversary with knowledge of specific sensitive attributes. So, there seems to be a diminishing value to these classical implementations of anonymization, and in closing we encourage the reader to always be open to better methods for anonymization - possibly differential-privacy, as a suggestion for a future area of exploration.

6 Conclusion

In this paper, we provided definitions for several anonymity models for health databases including naive-k-anonymous model, advanced-k-anonymous model, and Diverse-L-over-advanced-k-anonymous model. We have also researched several attacks on anonymised databases, and generated several games that the adversaries can play on anonymous databases. Additionally, several constructions of the database are given and are proved to have different levels of security under different levels of attacks, each with its own purpose as well as shortcomings.

References

1. Intersoft Consulting.: GDPR homepage, https://gdpr-info.eu/, last accessed 2021/11/2.
2. Iyiola, E., Jens, R., Matthias, K., Megha, K. A Review of Anonymization for Healthcare Data. Oxford University Press (2021).
3. Kern, M.: *Formalisation of privacy L-diversity*. Seminar paper, Technische Universitat Munchen, Munich (2013)
4. Latanya, S.: K-anonymity: A Model for Protecting Privacy. Carnegie Mellon University (2002).
5. Latanya, S.: Simple Demographics Often Identify People Uniquely. Carnegie Mellon University, Data Privacy Working Paper 3. Pittsburgh (2000).
6. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-Diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data 1, 1, Article 3 (2007).
7. Personal Data Protection Commission Singapore. Guide to Basic Data Anonymisation Techniques (2018)

AirTags or “AirTacks”? Spoofing attacks of Apple AirTags

Jet Kan Yip Keng, Nigel Ng , William Ryan Kusnadi, Wong Shun Min

National University of Singapore, Singapore 138607

Abstract. The Internet of Things(IoT) field has grown rapidly over the past few years. Many new IoT devices and systems have been invented and marketed by different companies. In April 2021, Apple released its newest IoT device called AirTags. Our paper will discuss the security aspect of AirTags and potential spoofing and data exfiltration on Apple’s newest IoT device.

1 Introduction

Apple introduced AirTags in April 2021 with the intention to help people track their belongings. It works in pairs with other Apple devices, such as Iphone, and users can use the Find My app functionality to get the location of their belongings with airtag attached to it. Apple uses a public key encryption scheme in the implementation of AirTags. We will briefly discuss how AirTags work in the next subsection.

1.1 How AirTags Work

A brief explanation of how AirTags work is as follows:

1. Pairing of AirTag with other Apple devices. During the pairing phase, a key pair will be generated. The public key will reside on AirTag, the private key will reside on the paired Apple device. An additional shared secret key will also reside on AirTag which is used to generate rolling public keys every 15 minutes.
2. Every 2 seconds, AirTag will send Bluetooth Low Energy broadcasts with the public key as its content.
3. The nearby Apple devices will recognize the Find My Device broadcasts and encrypt their current locations with the public key in the broadcast’s content and upload it to the Apple server.
4. When the AirTag’s owner wants to find out the location of the AirTag, the paired Apple device will generate a list of rolling public keys deterministically using the shared secret key generated earlier and query the Apple server to get the encrypted location.
5. The paired owner device then decrypts the encrypted location retrieved using the private key.

2

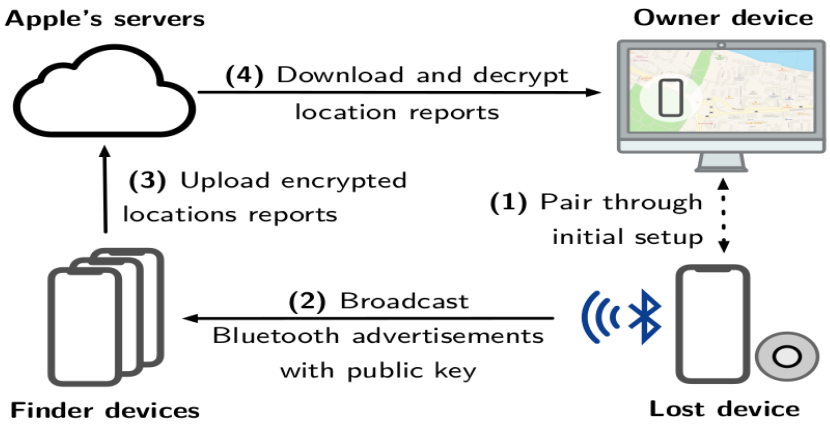


Fig. 1. Simplified workflow of AirTags [2].

More formally on how AirTags work is as follows,

Gen: during pairing phase, output public key(p_k), private key(s_k), and shared secret key(ss_k).
Gen_pk: every 15 minutes, on AirTag, replace public key with fresh public key generated using the shared secret key ss_k.
Enc: on input p_k and location, output the encrypted location c using ECIES[1].
Dec: on input s_k and encrypted location c, return actual location m.
Query: on input p_k, return a list of encrypted locations with p_k as the key.

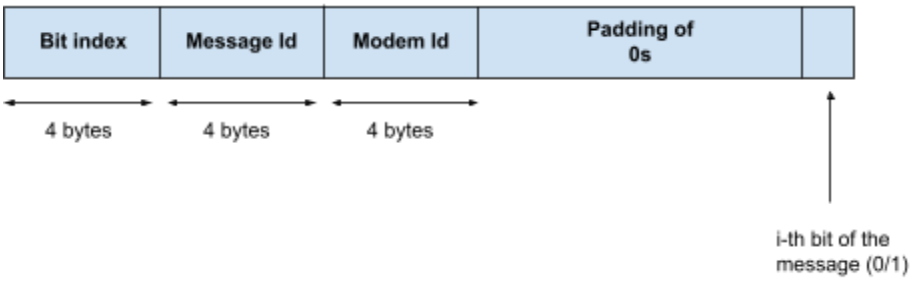
2 Data Exfiltration with AirTag

Looking at the mechanism of AirTags, it seems possible to upload arbitrary data to the Apple server and query them again. This creates a potential data exfiltration attack since any adversaries can store/retrieve sensitive data using the Apple server.

2.1 Implementation of data exfiltration protocol

A brief explanation of how the protocol is implemented:

1. To upload an entire message to the Apple server, we have to loop over all individual bits of the message and encode each of them into a 28-byte size array of the form:



2. We would then treat each of the 28-byte size constructed arrays as an AirTag public key and broadcast it using Apple's Find my network system. Any nearby Apple device that picks up the public key will encrypt its current location with the public key and upload it to the Apple server.

3. Next, we would attempt to fetch/receive the message that was previously stored in the Apple server. We reconstruct the message using a brute force method, which is by generating two 28-byte arrays for each bit index, each representing both possibilities for that guessing bit (0 or 1).

4. We then treat these two 28-byte arrays as two AirTag's public keys. By querying the Apple server using the SHA-256 hashes of these two public keys, only 1 hash should have valid location reports because it was stored previously in step (2). This infers the corresponding bit encoded by this 28-byte array in our original message.

5. Repeat the step (3) & (4) for each bit until the original message is fully reconstructed.

E.g Query Apple's backend using bit-index 1 of the message

2.2 Game definition

To test how reliably a PPT adversary A can abuse this protocol to transmit arbitrary data, we construct a game as follows:

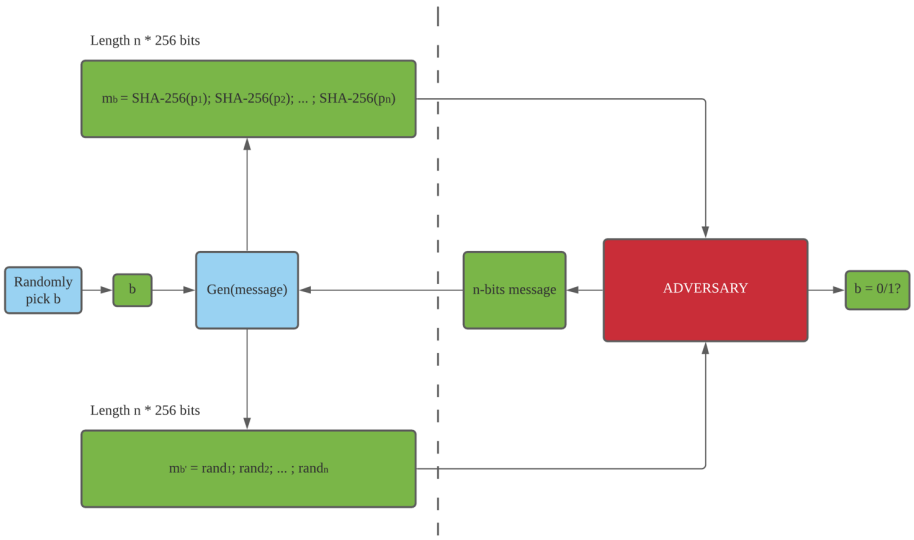


Fig. 3. Adversarial game for AirTag data exfiltration

1. The adversary A generates an arbitrary message of length n.
2. The challenger C uniformly picks b ∈ {0, 1}
3. C generates two messages of the same length, where m_b is the n sequences of SHA-256 hashes as the result of encoding the n-bit message using the data exfiltration protocol P and $m_{b'}$ is a sequence of uniformly random bits.
$$m_b = \text{SHA-256}(p_1); \text{SHA-256}(p_2); \ldots ; \text{SHA-256}(p_n)$$
$$m_{b'} = \{0, 1\}^{n * 256}$$
4. A wins the game if they can distinguish the encoded message and the random sequence from m_0, m_1. In such cases, we write $\text{Exfil}_{A,P}(n) = 1$.

2.3 **Calculating** $\Pr[\text{Exfil}_{A,P}(n) = 1]$

Assuming that A has the capability to encode the message, denoted as ANS, using the same data exfiltration protocol P as the challenger. After receiving m_0 and m_1 , A compares both of them with ANS. Since m_b is always identical to ANS, there are two cases:

1. $m_b = m_{b'}$. A wins ½ of the time by randomly choosing from m_b and $m_{b'}$
2. $m_b \neq m_{b'}$. A always wins by choosing m_b because it is the only input that is identical to ANS

Thus, the probability of A winning the game is

$$
\begin{aligned}
&\Pr[\text{Exfil}_{\mathcal{A},\mathcal{P}}(n) = 1] \\
&= \Pr[\text{Exfil}_{\mathcal{A},\mathcal{P}}(n) = 1 \wedge m_b = m_{b'}] + \Pr[\text{Exfil}_{\mathcal{A},\mathcal{P}}(n) = 1 \wedge m_b \neq m_{b'}] && \text{(joint probability)} \\
&= \Pr[\text{Exfil}_{\mathcal{A},\mathcal{P}}(n) = 1 | m_b = m_{b'}] \cdot \Pr[m_b = m_{b'}] \\
&\quad + \Pr[\text{Exfil}_{\mathcal{A},\mathcal{P}}(n) = 1 | m_b \neq m_{b'}] \cdot \Pr[m_b \neq m_{b'}] && \text{(conditional probability)} \\
&= \frac{1}{2} \cdot \Pr[m_b = m_{b'}] + 1 \cdot \Pr[m_b \neq m_{b'}]
\end{aligned}
$$

Since each bit in $m_{b'}$ is uniformly distributed, we have

$$
\begin{aligned}
\Pr[m_b = m_{b'}] &= \left(\frac{1}{2}\right)^{\texttt{len}(m)} \\
&= \left(\frac{1}{2}\right)^{n \cdot 256} \\
&= \text{negl}(n) && \text{(DEFINITION 3.4)} \\
\Pr[m_b \neq m_{b'}] &= 1 - \Pr[m_b = m_{b'}] && \text{(complement rule)} \\
&= 1 - \text{negl}(n)
\end{aligned}
$$

Applying $\Pr[m_b = m_{b'}]$ and $\Pr[m_b \neq m_{b'}]$ to $\Pr[\text{Exfil}_{A,P}(n) = 1]$,

$$
\begin{aligned}
&\Pr[\text{Exfil}_{\mathcal{A},\mathcal{P}}(n) = 1] \\
&= \frac{1}{2} \cdot \Pr[m_b = m_{b'}] + 1 \cdot \Pr[m_b \neq m_{b'}] \\
&= \frac{1}{2} \cdot \text{negl}(n) + 1 \cdot (1 - \text{negl}(n)) \\
&= 1 - \frac{1}{2} \cdot \text{negl}(n) \\
&= 1 - \text{negl}(n) \qquad \text{(PROPOSITION 3.6)}
\end{aligned}
$$

This shows that A is able to win the game almost 100% of the time. Therefore, we conclude that A can reliably use this data exfiltration protocol to store data in Apple server and retrieve by broadcasting fake AirTag public keys.

2.4 Loss of information during transmission

In part 3.3, we assume that there is no loss of information during the message transmission. In practice, this protocol can only reconstruct 23 out of 32 bits of the original message on average due to encryption errors, corrupted signals, etc. Using this information, we can modify how A distinguishes the inputs and update the $\Pr[\text{Exfil}_{A,P}^{\text{loss}}(n) = 1]$.

After receiving m_b and $m_{b'}$, A decodes these two messages and compares them with the original message. Let's denote the similarity between m_i and the original message

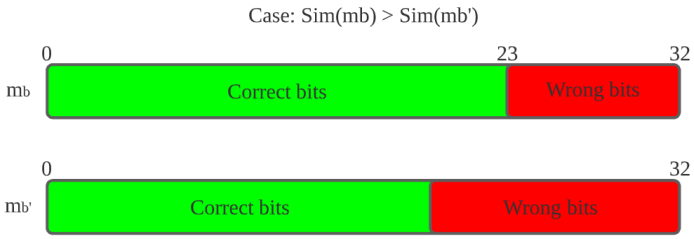
(i.e. number of correctly reconstructed bits) as $Sim(m_i)$. A always chooses the input that has the higher similarity score. There are three possible cases:

1. $Sim(m_b) > Sim(m_{b'})$. A always wins by choosing m_b
2. $Sim(m_b) = Sim(m_{b'})$. A wins ½ of the time by randomly choosing from m_b and $m_{b'}$
3. $Sim(m_b) < Sim(m_{b'})$. A always loses by choosing $m_{b'}$

Thus, the probability of A winning the game is

$$
\begin{aligned}
&\Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1] \\
&= \Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1 \wedge (\text{Sim}(m_b) > \text{Sim}(m_{b'}))] \\
&\quad + \Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1 \wedge (\text{Sim}(m_b) = \text{Sim}(m_{b'}))] \\
&\quad + \Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1 \wedge (\text{Sim}(m_b) < \text{Sim}(m_{b'}))] && \text{(joint probability)} \\
&= \Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1 | \text{Sim}(m_b) > \text{Sim}(m_{b'})] \cdot \Pr[\text{Sim}(m_b) > \text{Sim}(m_{b'}))] \\
&\quad + \Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1 | \text{Sim}(m_b) = \text{Sim}(m_{b'})] \cdot \Pr[\text{Sim}(m_b) = \text{Sim}(m_{b'}))] \\
&\quad + \Pr[\text{Exfil}^{\text{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1 | \text{Sim}(m_b) < \text{Sim}(m_{b'})] \cdot \Pr[\text{Sim}(m_b) < \text{Sim}(m_{b'}))] && \text{(conditional probability)} \\
&= 1 \cdot \Pr[\text{Sim}(m_b) > \text{Sim}(m_{b'}))] + \frac{1}{2} \cdot \Pr[\text{Sim}(m_b) = \text{Sim}(m_{b'}))] + 0 \cdot \Pr[\text{Sim}(m_b) < \text{Sim}(m_{b'}))] \\
&= \Pr[\text{Sim}(m_b) > \text{Sim}(m_{b'}))] + \frac{1}{2} \cdot \Pr[\text{Sim}(m_b) = \text{Sim}(m_{b'}))]
\end{aligned}
$$

To calculate $Pr[Sim(m_b) > Sim(m_{b'})]$, we assume that the m_b has 23 correct bits out of a total of 32 bits, while $m_{b'}$ has less than 23 correct bits out of 32 bits. We can visually represent the proportion of correctly reconstructed bits in green, and the incorrectly reconstructed (due to encryption errors, corrupted signals, etc.) bits in red as such



Thus the probability of $m_{b'}$ having less correct bits than m_b is equivalent to the probability of $m_{b'}$ having less than 23 correct bits, which is also equivalent to $m_{b'}$ having more than 9 incorrect bits.

$$
\begin{aligned}
\Pr[\text{Sim}(m_b) > \text{Sim}(m_{b'}))] &= \Pr[\text{Sim}(m_{b'}) < 23] \\
&> \frac{9}{32}
\end{aligned}
$$

On the other hand, the probability of m_b having the exact same number of correct/incorrect bits with $m_{b'}$ can be calculated using binomial theorem.

$$\begin{aligned}\Pr[\mathrm{Sim}(m_b) = \mathrm{Sim}(m_{b'}))] &= \Pr[\mathrm{Sim}(m_{b'}) = 23] \\ &= \binom{32}{23} \cdot \left(\frac{1}{2}\right)^{23} \cdot \left(\frac{1}{2}\right)^{9} \\ &= \binom{32}{23} \cdot \left(\frac{1}{2}\right)^{32}\end{aligned}$$

Thus, the probability of A winning the game is

$$\begin{aligned}&\Pr[\mathrm{Exfil}^{\mathrm{loss}}_{\mathcal{A},\mathcal{P}}(n) = 1] \\ &= \Pr[\mathrm{Sim}(m_b) > \mathrm{Sim}(m_{b'}))] + \frac{1}{2} \cdot \Pr[\mathrm{Sim}(m_b) = \mathrm{Sim}(m_{b'}))] \\ &> \frac{9}{32} + \frac{1}{2} \cdot \binom{32}{23} \cdot \left(\frac{1}{2}\right)^{32} \\ &> \frac{9}{32} + \binom{32}{23} \cdot \left(\frac{1}{2}\right)^{33}\end{aligned}$$

This shows that A is able to win the game with non-negligible probability.

3 Spoof Security of AirTag

In the case of AirTag, we want to know whether AirTag is spoof-secure, that is an adversary should not be able to determine which public key belongs to which AirTag device (in a multiple AirTags scenario). We design an adversarial game $\mathrm{PrivK}_{\mathrm{A}}^{\mathrm{spoof}}$ in order to approach the question.

8

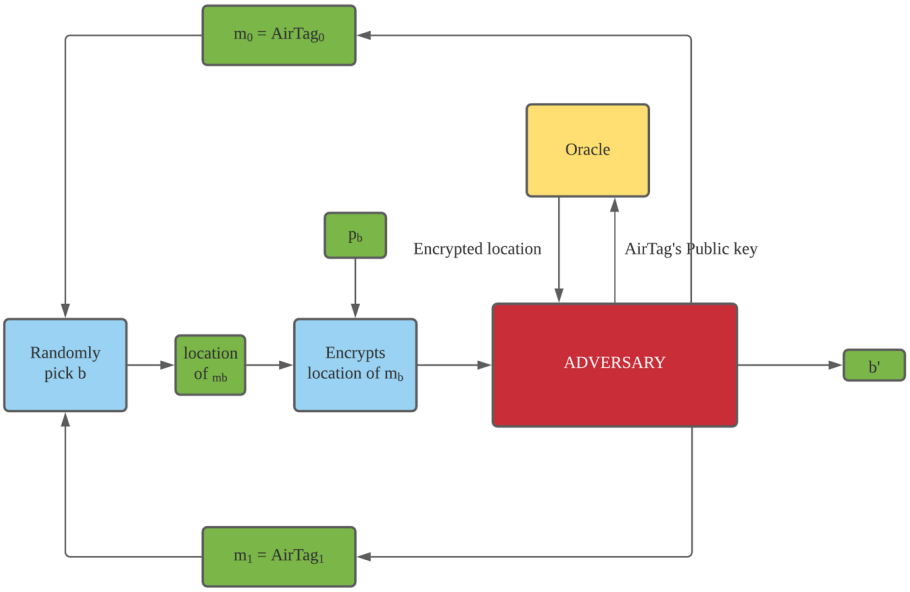


Fig. 2. Adversarial game for AirTag spoofing

1. Adversary chooses 2 AirTags devices m_0, m_1. Adversary knows the exact location for the 2 devices, but does not know both AirTag's public and private keys.
2. Challenger uniformly picks $b \in \{0, 1\}$. Then, the challenger encrypt the location of m_b with its public key p_b, $c \leftarrow Enc_b$(location of m_b).
3. Adversary will have access to Oracle which in this case is the Apple server. The oracle takes in any AirTag's public key and outputs the list of encrypted locations with that particular public key.
4. On input c, Adversary A outputs a bit b'.
5. A succeeds if b' = b. In such cases, we write $PrivK_A^{spoof} = 1$.

When we are considering real world scenario, there are some limitations on the adversary A as follows:

1. An adversary doesn't have unlimited time T to query the Oracle. With Apple AirTag implementation, the public key of an AirTag will be rolled every 15 minutes. Apple also might impose rate limiter logic to prevent excessive query.

Thus, with the practicality in mind, we define that AirTag is Non-Spoofable, if for all PPT adversary A, there exists a negl, such that,

$$\Pr[\mathrm{PrivK}_A^{spoof} = 1] \geq ½ + \mathrm{negl}(n)$$

3.1 Brute Force Attack on AirTag

Since an AirTag's location is encrypted with public key encryption scheme, using the AirTag's broadcasted public key, the only possible way for an adversary A to reverse engineer the public key using the AirTag's encrypted location report is by brute forcing the public key and querying the oracle for each public key. When an adversary does so, the probability of winning by brute force is 1/(2^n), where n is the length of the public key. According to definition 3.4, we say that an adversary has a negligible probability of winning the game by brute force. Therefore, the total probability of winning the game is ½ + negl(n) as b = {0, 1} is uniformly chosen.

4. What is on the horizon? Where are we going to be in 10 or 20 years time?

Currently, AirTags are being implemented with Bluetooth technology. These little smart devices were mainly designed to keep track of your own items, be it your house keys or even your pet dog. AirTags are equipped with Precision Finding, which allows the user to see an arrow on his/her iPhone's display and know exactly how close he/she is to the AirTag as well as the direction it is in (Peterson J, 2021). This technology could potentially be used in contact tracing, where the user can be precisely tracked with the AirTag. This can also allow deep cleaning to be done easily if the AirTags are able to store the exact location of places that it went to.

Another usage of an AirTag would be to use it as an NFC contact card. A user can tap an AirTag with a NFC-equipped smartphone and then be redirected to a webpage that displays the information which was provided (Peterson J, 2021). This brings up another usage for AirTags. It can be used as "secret message dead drops" (Peterson J, 2021). Two entities can place their AirTags at the same location, and then scan each other's AirTag to get the information provided by the other user.

AirTags can also be used in games like scavenger hunt (Peterson J, 2021). The players can use their iPhones to easily track the clues. However, it would be hard to determine which AirTag to navigate to since the iPhone will display multiple AirTags. As more features are being implemented for AirTags, there is a high possibility that AirTags might be featured in upcoming new games.

Some time ago, Apple has announced that AirTags will also support visual and audible cues, meaning that AirTags could be used in Augmented Reality (AR) applications. Some experts say that AirTags technology is already using ARKit (Goode L, 2021), which is a tool for producing AR experiences in applications or games. This could broaden the possibilities for the practical usage of AirTags. Like

many companies, Apple is also working on AR projects which include AR headset and glasses.

As AR's development continues to grow, it can change the way we use our devices. In the near future, AR glasses or headsets might be used as a substitute for smartphones. Everything done on smartphones will be done via motion and gesture detection, be it calling or texting someone and checking the weather forecast.

References

1. Antipa, Adrian., Brown, Daniel., Menezes Alfred., Struik, Ren'e., Vanstone , Scott.: Validation of Elliptic Curve Public Keys
2. Braunlein, Fabian. "Send My: Arbitrary data transmission via Apple's Find My network" Positive Security, 12 May 2021, https://positive.security/blog/send-my.
3. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016).
4. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999).
5. Author, F.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010).
6. LNCS Homepage, http://www.springer.com/lncs, last accessed 2016/11/21.
7. Peterson, J. (2021, April 29). *18 surprisingly practical uses for Apple AirTags*. Gadget Hacks. Retrieved November 7, 2021, from https://ios.gadgethacks.com/how-to/18-surprisingly-practical-uses-for-apple-airtags-0384571/.
8. Goode, L. (2021, April 21). *AirTags are the perfectly boring, functional future of Ar*. Wired. Retrieved November 7, 2021, from https://www.wired.com/story/apple-airtags-future-of-augmented-reality/.

Security Evaluation of Blockchain Technologies CS4236 Project

Balachandar Gowrisankar, Jiang Yuxin, Jonathan Foo, Kor Ming Soon, and Teo Jia Wei

National University of Singapore

Abstract. This paper attempts to evaluate the security of Blockchain Technology in a formal approach. It describes the increasing importance of blockchains in our daily lives and introduces the technical terminology used in the context of a blockchain. Security principles of blockchains are then formulated based on the current security practices of popular blockchains like Bitcoin. The principles are: $BlockContent - Secure$, $Trans - ForgeFree$, $Trans - CommitTime$ and $Time - Secure$. These principles are built on the traditional security properties - Integrity, Availability and Authenticity. This paper also provides mathematical construction of games for the security principles and proofs are given to show how a blockchain is secure with respect to these games. The conclusion of the proofs show how an attacker only has a negligible probability of winning the games. Nevertheless, there are existing attacks on blockchains such as the Eclipse attack and the 51% attack that violates the security principles of $Trans - ForgeFree$ and $Time - Secure$ respectively. These attacks are successful due to existing advantages that an attacker has (High CPU Power) or conducting side-channel attacks, allowing the attacker to bypass the security of the blockchain with respect to the relevant security principle. The security principles of the blockchain are then evaluated and its limitations are addressed.

Keywords: Blockchain · Transaction · Digital Signature · Merkle Tree · Transaction Time · BlockContent-Secure · Trans-ForgeFree · Trans-CommitTime · Time-Secure

1 Introduction

A blockchain is a shared and distributed ledger that facilitates secure transactions among untrusted parties in a peer-to-peer network [1,2]. It provides a decentralised network where parties no longer need a central authority to trust when transactions are made. Instead, cryptographic proofs are used for direct transactions between two parties where these transactions are computationally impractical to reverse. The Bitcoin blockchain, a cryptographic-based digital currency system, was first published by Satoshi Nakamoto in 2008 [3]. Other blockchain applications including Ethereum and Hyperledger Fabric have also been developed since then [2]. Recently, blockchain has been recognised as a

promising technology and is growing in popularity in various industries including Internet of Things, healthcare, banking and logistics due to its properties of decentralisation and trustlessness [1,2]. As the usage of blockchains becomes more prevalent, there is a need to ensure its security as well. In this paper, we attempt to formalize the security of blockchain technologies by constructing and evaluating security principles for a blockchain. Games are also created to formally prove the security of a blockchain with respect to the security principles. In the next section, the technical components of blockchain will be introduced as these concepts will be important in understanding the formulation of the security principles.

1.1 Structure of a Blockchain Block

A blockchain database keeps valid transaction records in blocks which cannot be deleted or modified by users [1]. Each block consists of a block header and a block body [4]. Figure 1 in appendix 1 illustrates the structure of blockchain blocks.
A block header includes [1,4]:

1. **Nonce**: a value for hash calculation that miners are going to solve
2. **Merkle tree root hash**: a hash value of all transactions which are organized in a Merkle structure in the block
3. **Data**: transaction information including amount of money to be transferred
4. **Parent block hash**: hash value of the previous block
5. **Timestamp**: block creation time and hash,

The block body consists of a transaction counter and transactions.

1.2 Blockchain Network

A blockchain network consists of blockchain nodes which are able to create and validate transactions and perform mining [5]. Blockchain users create transactions by broadcasting them to the network and perform mining to create new blocks. Miners compete to create valid blocks to earn incentives. For example, miners receive Bitcoins for successfully mining a Bitcoin block.
There are various consensus algorithms that determine which miners win rewards as well as ensure the consistency of ledgers among different blockchain nodes. These algorithms include Proof of Work, Proof of Stake, Proof of Space etc [1].

- **Proof of work (PoW)**: Miners need to solve a complex mathematical problem to submit a block into a blockchain. The new block is validated by other nodes in the network before it can be appended to the blockchain and the miner can receive rewards [1]. Bitcoin uses PoW as its consensus algorithm.

– **Proof of stake (PoS)**: Stakers who put up more currency as stake are more likely to be selected as the next validator. Compared to PoW, PoS saves energy but gives little reward to the validator [1].

– **Proof of space (PoSpace)**: Miners are selected based on their capability of high space [4].

1.3 Past Attacks

In this section, successful attacks on different blockchains are introduced. Whilst the attacks are not explained in detail, it is worth noting that some of the security principles formulated in the next sections take inspiration from these attack vectors.

One instance of a blockchain attack is the distributed denial-of-service (DDoS) attack that caused Bitfinex to suspend its service temporarily in 2017 [4]. Additionally, in May and June 2018, a 51% attack on Monacoin, Bitcoin Gold, Zencash, Verge, and Litecoin Cash caused a 5 million USD loss [6]. Apart from the above-mentioned attacks, the following attacks explore the different types of exploit:

1. **Selfish Mining**:
   Instead of releasing blocks to the public, attackers deliberately keep blocks private in order to produce a longer chain than the public chain. Selfish miners obtain rewards by releasing their blocks when public chain length gets close to the private chain's length [6].

2. **The Majority Attack**:
   Also known as a 51% attack, attackers who obtain over 50% of hash rate are highly likely to append their blocks to the chain successfully [6].

3. **Distributed Denial of Service Attack**:
   DDoS attacks are performed in various ways. For instance, 51% attack can cause denial of service in Bitcoin. 51% attack prevents other miners from appending blocks to blockchain and invalidate current transactions, which leads to failure of service in the network [6].

2 Related Works

With the increased interest in Blockchain technology in recent years, many surveys as well as research papers have been published reviewing the benefits and vulnerabilities of Blockchain. For example in [1], the Blockchain is broken down in the various layers and further analysed. Another example [2] explores in deeper detail the varying cryptographic approaches behind different blockchain platforms and the challenges of ensuring privacy, integrity and availability. Both papers also suggested possible solutions against the present challenges.

As illustrated above, many papers explore the vulnerabilities of Blockchain technology, the attacks on it as well as solutions against possible attacks. However, we felt that we would add little value by reiterating what other papers have already covered but rather leverage our CS4236 knowledge to challenge the underlying primitives of Blockchain security. We do this by introducing security principles and constructing games and proofs for these principles in the later sections.

3 Formulation of Blockchain Security Systems

3.1 Blockchain Hash System:

For the Blockchain Hash System $(\mathcal{P}, \mathcal{V}, H)$, we define the following two entities and a hashing algorithm, H:

1. Prover $\mathcal{P}$: is an entity who wants to convince the verifier $\mathcal{V}$ that it owns all the data.
2. Verifier $\mathcal{V}$: is an entity who determines whether prover $\mathcal{P}$'s claim as the owner of the data is correct.
3. H takes as input a string x and outputs a hash string $H^s(x)$

3.2 Blockchain Digital Signature System:

We define a blockchain digital signature system $(Gen, Sign, Vrfy)$ that compromises of 3 algorithms:

1. $Gen(1^n)$: The input is the size of the key and the output is (p_k, s_k).
2. $Sign_{s_k}(t)$: Input is the private key s_k and the transaction t. Output is the signature s.
3. $Vrfy_{p_k}(s, t)$: Input is the public key p_k and the signature-transaction pair (s, t). Output is a bit b. If $b = 1$, this means that the signature is valid. Else, when $b = 0$, the signature is invalid.

4 Security Principles of Blockchain

In this section, we attempt to define security principles that aim to keep Blockchain secure. This principles are built on the fundamental security requirements such as Integrity, Authenticity and Availability. These principles aim to defend a blockchain network against possible attack vectors.

4.1 BlockContent-Secure

The traditional integrity verification of using MAC algorithms is an overwhelming process and requires a relatively larger time to compute. Practically, the computation also consumes more bandwidth. With this limitation, blockchain depends on a lighter method of integrity verification: Hashing.

Blockchain ledgers comprise blocks of data in a fixed order. Each block is linked to the one before with a hash pointer. This hash pointer is essentially a hash of the properties of the block, containing the block prior to it as well its own information such as transactions and the proof-of-work. The final hash value of the latest block will be used to verify the integrity of the entire chain. Figure 1 in Appendix 1 shows the Blockchain with the Hash value inside.

The hash mechanism that most blockchains adopt is the Merkle Tree hash. The introduction of the Merkle Tree hash in the verification process of the blockchain not only reduces the computational processes from $\mathcal{O}(n)$ to $\mathcal{O}(log\ n)$ (where n is the number of transactions), but also helps to maintain integrity of the data in an efficient manner. In figure 3, we provide an illustration of the Merkle Tree.

We now define a $BlockContent - Secure$ as an ideal security property of a blockchain. For a blockchain to be $BlockContent - Secure$, it must be impossible for a block to be modified and still be accepted by the entire blockchain ledger. However, what we shall explore is how the authentication of the Merkle Tree hashing is completed in blockchain. The authentication in the construction and verification of the Merkle Tree is paramount in its role to ensure the integrity of the Blockchain.

Construction of BlockContent-Secure Game:

Definition 1. *In our following game $BCS_{A,H}$, we assume that the primitive H used to construct the Merkle Tree is a hash function that meets the requirement of pre-image and collision resistance. Refer to Figure 4 in Appendix 1 for the illustration of the game.*

1. *The adversary A is given input 1^n and the hash function H to construct a Merkle Tree.*
2. *A sends a request to the Prover Oracle by providing an arbitrary n bit string. In turn, the Prover Oracle generates a transaction D' and constructs a Merkle Tree root value, Proof(D') (with pre-existing transactions as siblings in the Merkle Tree). The Prover Oracle then sends the Proof(D') to A. We note that the generated transaction D' is shared between the two Oracles (Prover and Data).*
3. *A then sends a request to the Data Oracle by providing Proof(D'). In turn, the Data Oracle then sends A a response of D'.*

4. *The challenger then generates a random transaction D, and generates the Merkle Tree root value as well, Proof(D), with the hash function H. The challenger then generates another string of random bits, Proof().*
5. *The challenger then randomly chooses random bit* $b \in \{0, 1\}$*, where* $proof_1$ *is Proof(D) and* $proof_0$ *is Proof(), and sends* $y = proof_b$ *to A.*
6. *A guesses the value of b upon receiving y, and outputs a value b'* $\in \{0, 1\}$*. If* $b = b'$*, we determine that the adversary A, wins the game.*

Definition 2. *We define further that for a block to be BlockContent-Secure, it cannot be modified iff for any adversary, there is a negl s.t:*

$$Pr[BCS_{A,H}(n)] \leq \frac{1}{2} + negl(n)$$

The proof for the above authentication protocol for the Merkle Tree construction can be rather trivial. Even with the accesses to the two Oracles, A will not be able to distinguish if the $proof_b$ comes from a valid transaction. This is because of the underlying H function, which ensures pre-image resistance. As a result, the adversary A will not be able to determine D. Therefore, $Pr[BCS_{A,H}(n)] \leq \frac{1}{2} + negl(n)$ holds.

4.2 Transaction Forgery Free

The fake identity is a classic issue in Bitcoin and many other blockchains: you pretend to be someone else and act in your own interests [8]. For instance, if Mary could steal Max's identity, she could initiate the transaction "Max sends 10 bitcoins to Mary". Hence, to prevent any forgery from happening, blockchains use digital signatures schemes to ensure the authenticity of a transaction.

Each transaction is signed by the user who initiates it. Anyone on the network can verify the transaction by using the user's public key. This prevents a malicious attacker from modifying a transaction without corrupting the entire block. Figure 5 illustrates a general idea of the usage of a digital signature in the Bitcoin blockchain. Furthermore, it allows for non-repudiation as it means that the user who sent the transaction had to have the private key and therefore owns the currency involved in the transaction.

We now define a $Trans - ForgeFree$ (Transaction Forgery Free) as a security property. For a transaction to be $Trans - ForgeFree$, it is impossible for an attacker to forge the signature and create a valid transaction under the identity of another user. This means that no attacker should be able to use the private key of other nodes in the network and sign using their private key. A node in a blockchain network can only sign a transaction using the node's own private key.

Construction of $Trans - ForgeFree_{A,\Pi}$ Game

Theorem 1. *Given a Blockchain Digital Signature Scheme scheme $\Pi = (Gen, Sign, Vrfy)$ and an adversary, consider the following game on transaction forgery:*

1. *$Gen(1^n)$ is executed to obtain the public and private keys (p_k, s_k) respectively. p_k and s_k are output from a uniform distribution with length n.*
2. *Adversary A is only given p_k and has access to an oracle O who has access to s_k.*
3. *The adversary can input any transaction t to the oracle O and the oracle O will output its signature $s = Enc_{sk}(t)$.*
4. *The transaction $< t >$ will then be stored in a database Q.*
5. *As the adversary A has access to the public key p_k, A can easily unsign the signature and perform $Vrfy_{p_k}(s, t)$ and check if the bit received is 1 or 0.*
6. *The adversary only wins if A can output a valid and new transaction and signature $< t_i, s_i >$ where $< t_i > \notin Q$.*

Figure 6 illustrates the $Trans - ForgeFree$ game.

Definition 3. *We define a term Trans-ForgeFree and say that a scheme Π is Trans-ForgeFree under a chosen-transaction attack iff for any adversary A, there is a negl s.t.*

$$Pr[Trans - ForgeFree_{A,\Pi}(n) = 1] \leq negl(n)$$

It is trivial to prove that this construction holds true. It is important that we assume that the generation of (p_k, s_k) to be secure. For example in the Bitcoin blockchain, the Elliptic Curve Digital Signature Algorithm (ECDSA) is used to generate the pair of keys.

Hence, under a secure digital signature system, the attacker's best choice is to randomly guess the private key s_k of the scheme. This results in a probability of $\frac{1}{2^n}$ since there are 2^n possible choices to choose from, implying that the probability is negligible.

4.3 Bounded Transaction Commit Time

There are two key characteristics of blocks produced in a blockchain:

1. Blocks are produced at (roughly) fixed intervals (e.g. 10 minute intervals in the Bitcoin blockchain).
2. Blocks have a fixed size.

When demand for block space exceeds the block space (i.e. size of transactions $>$ size of a block), the blockchain needs to have a mechanism to choose which transactions are included in the current block.

We define $TCT - Secure$ (Transaction Commit Time Secure) as an ideal security property of a blockchain. For a blockchain to be $TCT - Secure$, any user should be able to commit a transaction to the blockchain in bounded time, i.e. not have to wait an arbitrarily long time for their transaction to be committed.

Trans-CommitTime Game

Theorem 2. *We formulate a game where adversary A wins if A is able to generate a valid (m, t, mac) for an arbitrary timestamp t.*

1. *Generate key k = Gen(1^n).*
2. *A sends message m_i to the oracle. Oracle outputs (m_i, t_i, mac_i), where t_i is the timestamp that the oracle created this output and mac_i is the tag for the message (m_i, t_i).*
3. *A outputs (m, t, mac).*
4. *A wins (i.e. output of game is 1) iff (m, t, mac) is valid and (m, t, mac) $\notin$ Q.*

Figure 7 in the appendix illustrates the Trans-CommitTime game.

Definition 4. *We say that a blockchain scheme Π is TCT-Secure iff for any A, there is a negl s.t.*

$$Pr[TCT_{A,\Pi}(n) = 1] \leq negl(n)$$

Proof of TCT-Secure: We make use of Definition 4.3 from the lectures to prove that the blockchain scheme proposed is secure. We can conclude that an adversary A cannot generate a transaction with an arbitrary timestamp.

A blockchain scheme can make use of this property to utilise timestamps as a method to order transactions. The blockchain can guarantee that transaction T_i, sent at timestamp t_i, will be committed to the blockchain before transaction T_{i+1}, sent at timestamp t_{i+1}, where $t_{i+1} > t_i$ (i.e. t_{i+1} is later than t_i).

Based on this guarantee, a user is assured that by sending a transaction, it will be committed to the blockchain before another transaction that was sent at a later time, hence bounding the commit time of the transaction and thus fulfilling the Transaction Commit Time property.

Double Spend Attack : An attack on the TCT-Secure principle
There are certain blockchains which select transactions to be included in the next block based on the transaction fee of the transaction instead of the timestamp of the transaction. These blockchains are hence insecure w.r.t the $TCT - Secure$ principle and are susceptible to front running attacks [10]. Front-running attacks take advantage of the process of adding transactions to blocks based on transaction fees [10]. An attacker has the ability to ensure that their transaction is processed before any other transaction by including a higher transaction fee with it. This potentially leads to starvation of honest transactions as they will never be added to the network if the attacker constantly sets a higher transaction fee for their transactions than the honest transactions, violating the $TCT - Secure$ principle.

4.4 Time-Secure

Before we dive into the security formulation for $Time - secure$, let us see how all the nodes in a blockchain network come to a consensus of transaction blocks. The blockchain mechanism requires that all the nodes in the network should maintain the same copy of blocks/transactions. This prevents nodes from maliciously inserting transaction blocks into the network without the knowledge of the other nodes. However, consider this scenario. Let us say, we have four users in the blockchain network namely, Alice, Bob, Charlie and David. Suppose Alice wants to insert a fake transaction: "Bob pays Alice $10" into the network. Consider this as the attacker chain. Now, instead of broadcasting this to the entire network, Alice might just send it to Bob. But, Bob might be receiving blocks from Charlie and David that conflicts with the one sent by Alice. Let us consider this as the honest chain. The blockchain is modeled in such a way that when a node receives conflicting blocks, it should not immediately add it to its ledger. It must wait for few more blocks to be generated in that chain. It should then select the chain of blocks that is the longest. So for Alice to succeed, she must create more blocks with fake transactions, compute the proof of work for all of them and send them to Bob. She must also do this quickly because Charlie and David will also be sending blocks to Bob. So, essentially Alice has to race against time to create more blocks in the process of ensuring that her chain of blocks would be longer than the chain of blocks generated by Charlie and David.

We now define $Time - Secure$ as a security property. For a blockchain to be $Time - Secure$, it must be impossible for an adversary to create an alternate chain faster than the honest chain. If an attacker succeeds in creating a chain of blocks that is longer than the honest chain, then the blockchain is not $Time - Secure$.

Construction of Time-Secure $TS_{A,\Pi}$ Game
In this section, we formulate an experiment/game for Time-Secure principle. Let us consider that a blockchain network consists of four users: Alice, Bob, Charlie and David. Let us suppose that Alice plays the role of an attacker and the rest of the users are honest. All the four users are given the same chain of blocks to begin with. These four users will be adding blocks to that chain and the new chain will be sent to an Oracle. If the Oracle receives conflicting blocks for a chain, then it will store a copy of both the chains and will select whichever chain is the longest after a period of time. For simplicity, let us consider that the Oracle will select whichever chain is the first to get extended by 10 blocks from the point of receiving the conflicting block. If the Oracle ends up selecting the attacker chain, then we say that the attacker (i.e Alice) wins the game. So, if Alice wants to create a fake block in a chain, then she can win the game if she is able to create 10 additional blocks in that chain faster than Bob, Charlie and David. This is illustrated in Figure 8 in Appendix 1.

Formally, let π represent a blockchain network where the current block in the chain is B_0. Let A represent an adversary node who wants to inject malicious transactions into the blockchain network. We define an experiment $TS_{A,\pi}$ for Time-secure principle in blockchain as follows:

1. *The adversary A creates a fake block B_1' and sends it to the Oracle.*
2. *All the honest nodes try to compute the proof of work for the transactions in the network and create block B_1. This is sent to the Oracle.*
3. *The Oracle would now receive conflicting blocks for the chain following B_0. It will wait for the chains to get longer by 10 more blocks. Here, we are choosing the number '10' for simplicity. In reality, this number would be arbitrary.*
4. *The adversary and honest nodes keep working to create more blocks to add to their respective chains. As and when the blocks are created, they are sent to the Oracle.*
5. *When the Oracle receives 10 blocks in any chain, it outputs a bit 0 or 1. The output of the experiment is said to be 1 if the attacker chain is selected and 0 otherwise. We write $TS_{A,\pi} = 1$ if the output of the experiment is 1 and in this case we say that A succeeds.*

Definition 5. *A blockchain network π is Time-Secure under an attack iff for any adversary A, there is a negl s.t.*

$$Pr[TS_{A,\pi} = 1] \leq negl(n)$$

Proof of Time-Secure: The probability of an attacker winning the Time-Secure experiment is analogous to the Gambler's ruin problem [12]. Let us consider that the attacker is trailing the honest chain by z blocks. Then the probability that the attacker can catch up with the honest chain is as follows [3]:

$$Pr[TS_{A,\pi} = 1] = \begin{cases} 1, & \text{if } p \leq q \\ (q/p)^z, & \text{if } p > q \end{cases} \tag{1}$$

Here, p refers to the probability that an honest node finds the next block and q refers to the probability that an attacker finds the next block. We can see that the probability of success for the attacker drops exponentially as z increases.

Double Spend Attack: An attack on the Time-Secure principle
There are various versions of the Double spend attack which violates the Time-Secure principle. One such version is the 51% attack. One of the main requirements for this attack to succeed is that the attacker must be capable of controlling more than 50% of the network's computing power. If the attacker controls more than half the computing power of the network, then the chances of the attacker succeeding the Time-Secure experiment can increase. More details about this attack have been added in Appendix 3.

5 Evaluation

In the above section, the security principles of Blockchain are formulated and games are constructed based on these properties. It is desirable for a blockchain to possess all these security principles to ensure its integrity, authenticity and availability. This section summarises the four security principles and describes their limitations.

5.1 Summary of Security Principles

1. $BlockContent - Secure$: With the $BlockContent - Secure$ security property, a block's content cannot be modified maliciously by an attacker as the blockchain network will detect the corrupted block and remove it. This is mainly due to the usage of the Merkle Tree Hash as a hash function to prevent modifications in a block. Hence, the integrity of the block in the network is preserved.
2. $Trans - ForgeFree$: With the $Trans - ForgeFree$ security property, a transaction cannot be forged in the blockchain network. An attacker will not be able to impersonate another user on the network and create a transaction based on the other user. A blockchain is $Trans - ForgeFree$ when a secure digital signature scheme is used. This is because any transaction is signed using the private key of the sender and only the sender should be aware of his own private key. Hence, the authenticity of a transaction in the network is preserved.
3. $TCT - Secure$: With the $TCT - Secure$ security property, a user on the network will be assured that his transaction will be committed to the blockchain network in bounded time and an attacker is unable to make the user wait an arbitrary amount of time before the transaction is committed to the blockchain.
4. $Time - Secure$: With the $Time - Secure$ security property, it should be infeasible for an adversary in the blockchain network to add malicious blocks to the blockchain. This is because whenever a node receives conflicting blocks in a chain, it will wait for a while to select the chain that becomes longer after a period of time. The probability for an attacker to succeed in this case drops exponentially as the number of blocks he/she lags behind the honest chain increases. Hence, the availability of a transaction in the network is preserved.

5.2 Limitations of Security Principles

The four security principles formulated provide a high level overview of how the security of a blockchain can be modelled. It is important to note that the games constructed only ensures the security of a blockchain with respect to that security principle.

Due to the large scope of blockchain technology and its ever-increasing complexity, it is impossible to cover all possible attack vectors and their consequences using only 4 security principles. Furthermore, some real-world attacks on blockchains are based on side-channel attacks or attacks on the server hosting the blockchain network. These attacks bypass the security principles formulated for blockchains. One example of this is the Eclipse attack, which can be initiated by conducting a DDoS attack on a victim by flooding the victim with the attacker's IP address and forcing the victim to connect to the attacker-controlled nodes upon the restart of the victim software [11]. More details can be found in Appendix 3.

Whilst the security principles formulated for blockchain are based on fundamental security requirements, the security of a blockchain can also be viewed from a different perspective. One such perspective will be looking at blockchain security through the layers of a blockchain, namely the Application layer, Consensus layer, Network layer and Data layer. More details on these layers can be found in Appendix 2 as well.

6 Conclusion

In conclusion, as a shared and distributed ledger, blockchains enable secure transactions among parties who distrust each other in a peer-to-peer network. This paper provides a high level overview of security for a blockchain. Four blockchain security principles were formulated and games/experiments were defined. These security principles were then evaluated and their limitations addressed. In the near future, we find that potential for blockchains to be used in IOT is vast due to the decentralisation property of blockchains. Artificial Intelligence is another area where the data is intertwined to Blockchain. This creates a possibility where we can actually valuate data.

References

1. Salman, T., Zolanvari, M., Erbad, A., Jain R., Samaka, M. (2018). Security Services Using Blockchains: A State of the Art Survey. Retrieved November 1, 2021, from https://arxiv.org/pdf/1810.08735.pdf
2. Taylor, P. T. Dargahi, A. Dehghantanha, R. Parizi and K. Choo. (2020, May) A systematic literature review of blockchain cyber security, Digital Communications and Networks. Retrieved November 1, 2021, from https://doi.org/10.1016/j.dcan.2019.01.005
3. Nakamoto, S. (2009) Bitcoin: a peer-to-peer electronic cash system. Retrieved November 1, 2021, from http://www.bitcoin.org/bitcoin.pdf.
4. Zheng, Z., Xie, S., Dai H., Chen, X., Wang H. (2017, June 25) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends Retrieved. November 1, 2021, from http://doi.org/10.1109/BigDataCongress.2017.85
5. Tuyisenge, M. J. (2021, June 1) Blockchain technology security concerns: literature review. Retrieved November 1, 2021, from https://www.diva-portal.org/smash/get/diva2:1571072/FULLTEXT01.pdf
6. Saad, M., Spaulding, J., Njilla, L.L., Kamhoua, C.A., Shetty, S., Nyang, D., & Mohaisen, A (2019, April 6) Exploring the Attack Surface of Blockchain: A Systematic Overview. ArXiv, abs/1904.03487, 2019. Retrieved November 1, 2021, from https://arxiv.org/pdf/1904.03487.pdf
7. G Mcshane (2021, October 21) What is 51% attack. Retrieved November 1, 2021, from https://www.coindesk.com/learn/what-is-a-51-attack/
8. Thiery. (2020, May) How to represent a Blockchain through a mathematical model? Retrieved November 1, 2021, from https://canopee-group.com/wp-content/uploads/2020/05/Blockchain-Coperneec.pdf
9. Yakovenko, A. (2020, May 6). Proof of History: A Clock for Blockchain. Medium. Retrieved November 2, 2021, from https://medium.com/solana-labs/proof-of-history-a-clock-for-blockchain-cf47a61a9274
10. Behnke, R. (2021, July 5). What is a Front Running Attack? Retrieved November 6, 2021, from https://halborn.com/what-is-a-front-running-attack
11. Behnke, R. (2018, June 7). What is an Eclipse Attack? Retrieved November 6, 2021, from https://www.radixdlt.com/post/what-is-an-eclipse-attack
12. Columbia University (n.d.). Gambler's Ruin Problem Retrieved November 6, 2021, from http://www.columbia.edu/ ks20/FE-Notes/4700-07-Notes-GR.pdf
13. J Frankenfield (2021, August 25) 51% attack. Retrieved November 6, 2021, from https://www.investopedia.com/terms/1/51-attack.asp

7 Appendix 1

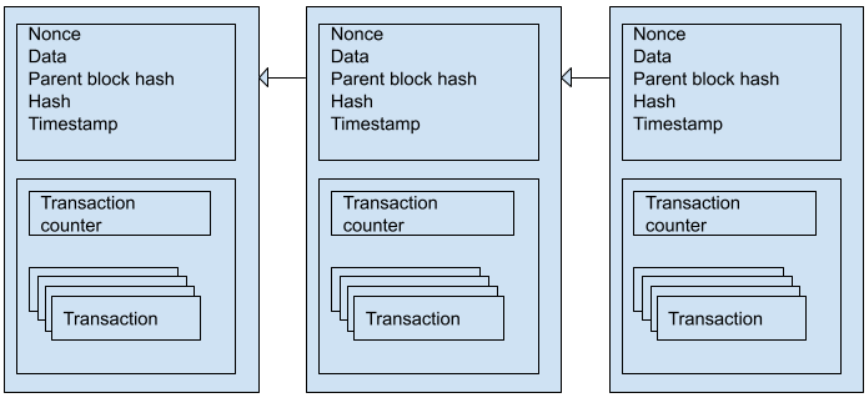


Fig. 1. Illustration of Blockchain Blocks

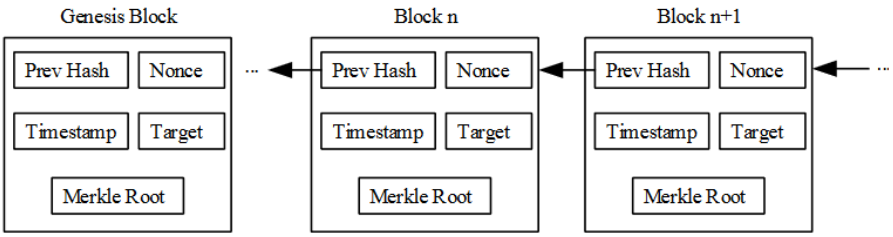


Fig. 2. Illustration of a chain of transaction

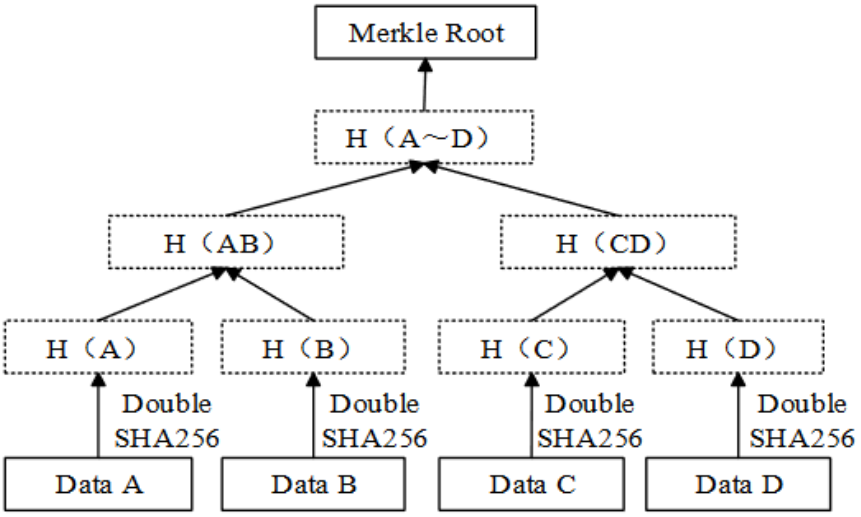


Fig. 3. Illustration of Merkle Root Hash

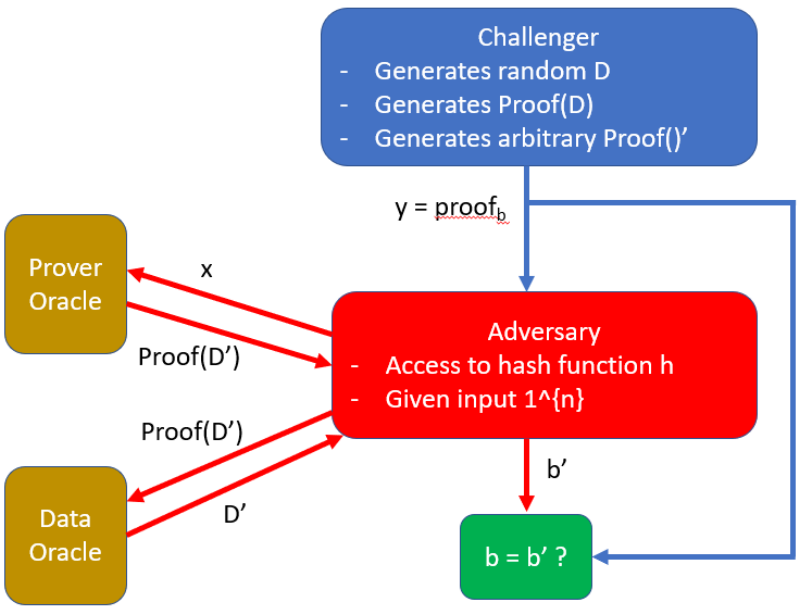


Fig. 4. Illustration of Merkle Tree Game

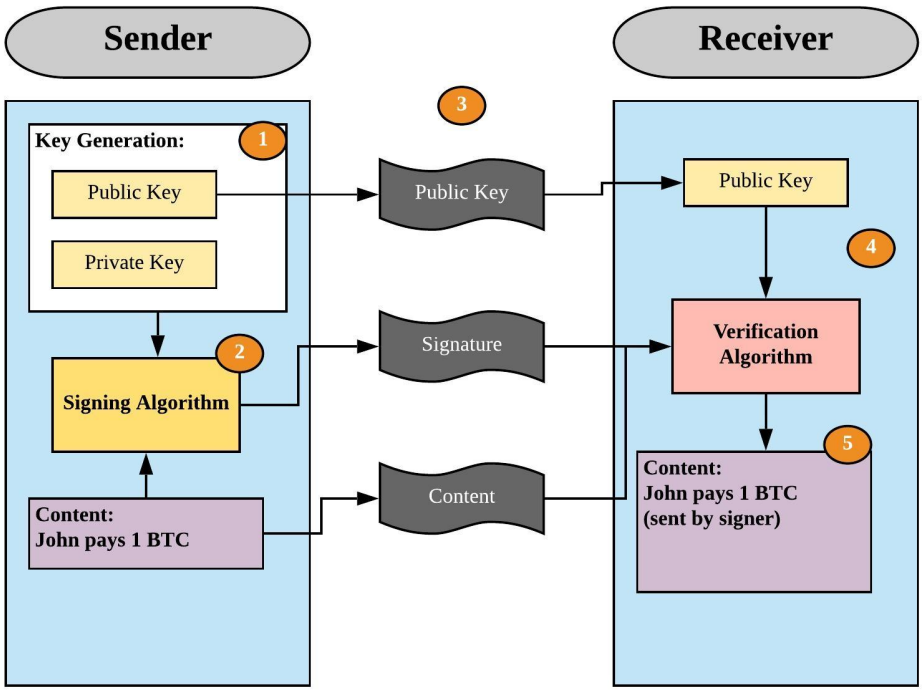


Fig. 5. Signing and verification of a transaction

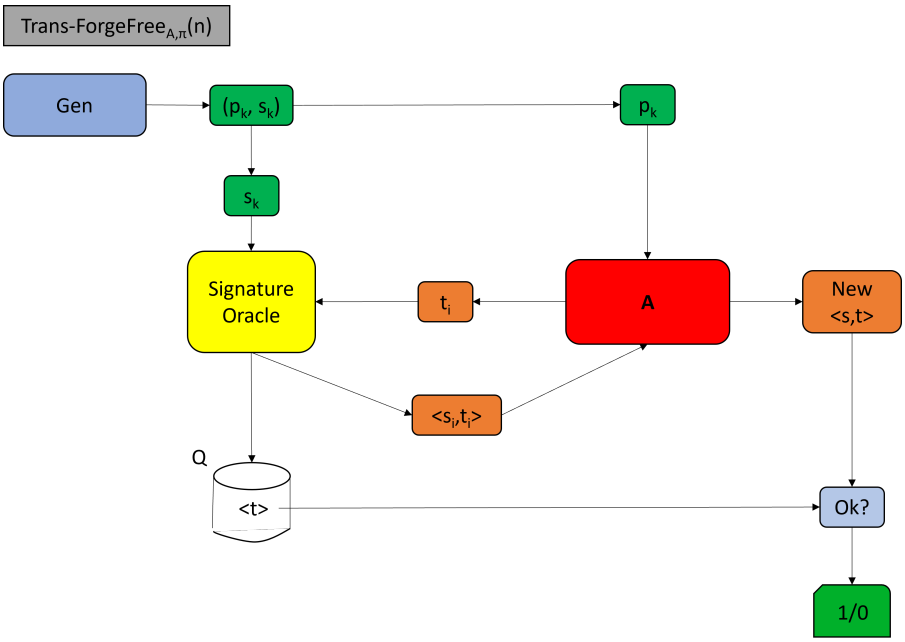


Fig. 6. Illustration of Trans-ForgeFree Game

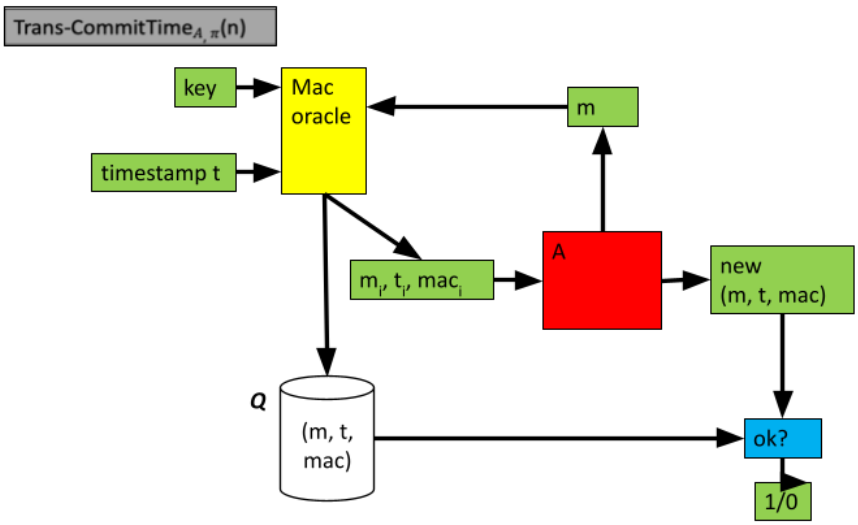


Fig. 7. Illustration of Trans-CommitTime Game

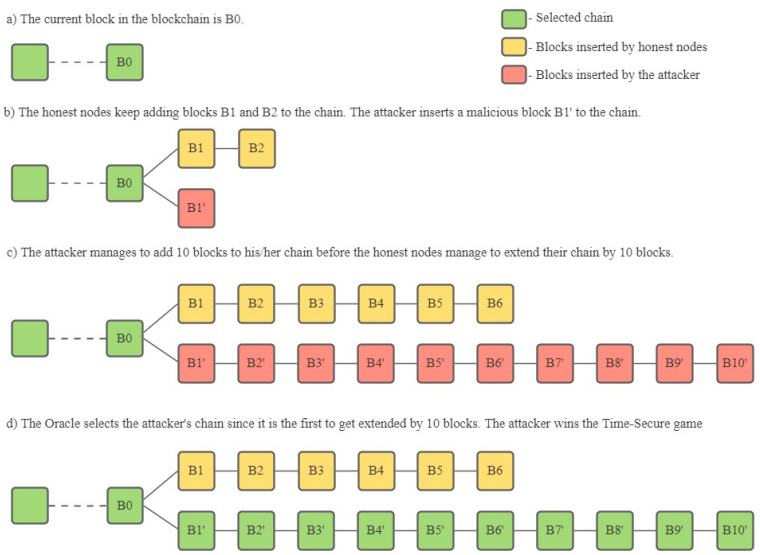


Fig. 8. Illustration of the Time-Secure game

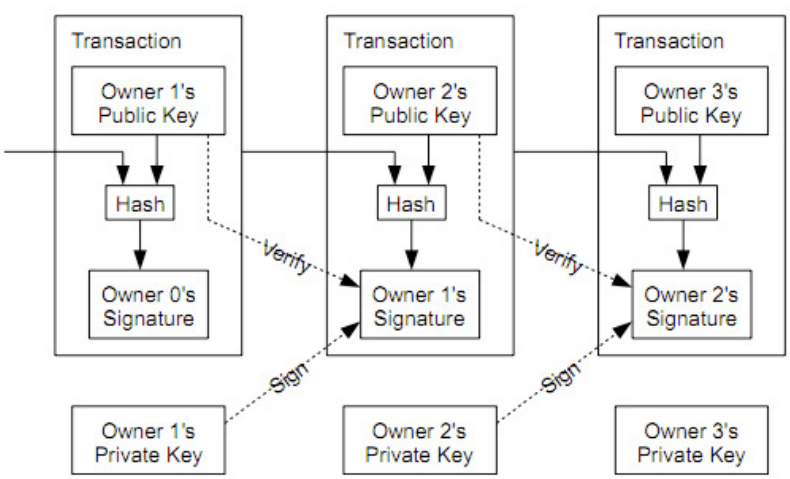


Fig. 9. Illustration of a chain of transaction

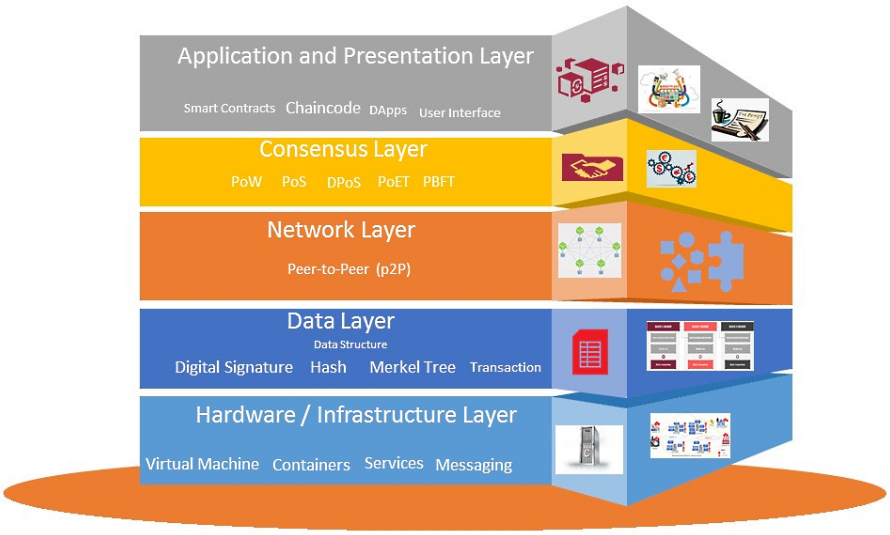


Fig. 10. Illustration of Blockchain layers

8 Appendix 2

8.1 Technical Terms of Blockchain

This section briefly lists out the basic and essential terms of blockchain to aid readers in understanding the terms used.

1. **Transaction:** Each transaction T consist of information related to the payment and parties. It can be defined as:

$$T = (previous\_transaction\_id, new\_owner\_id, \\ newowner\_publickey, amt\_transferred)$$

   The transaction is "signed" by the owner of the previous transaction. When a user claims that he owns the transaction T, he has to demonstrate that he knows the private key of p, where p is the public key of the owner. The owner of the transaction T hence the Block. Figure 9 in Appendix 1 shows a chain of transaction.
2. **Ledger:** A ledger is a history of transactions.
3. **Entry:** An entry is a transaction between two entities. We use the term Tr to denote a generic entry.
4. **Block:** A block B is defined as a vector of entries. Let nb be the size of a block.

$$B = (Tr_1, Tr_2, ..., Tr_{nb})$$

5. **Signature:** Blockchain uses Public-Key Cryptography and digital signature to prove who made the transaction and also makes sure that the content of the transaction is not modified.

6. **Proof-Of-Work:** A Proof-Of-Work is designed in a way that is very expensive to compute. We can think of it as a mathematical problem whose purpose is to create a link between two blocks. This link will be materialised within the header of the second block.

7. **Miner:** Someone who is trying to workout the proof-of-work is called a miner.

8.2 Core Layers of Blockchain

From a security perspective, there are 4 core layers in a blockchain: Application layer, Consensus layer, Network layer and Data layer. Each layer has the following responsibilities:

- Application layer is the top layer which enables usage by the user. Consensus layer consists of consensus algorithms for decision making.

- Network layer assures communication and synchronization between nodes in the network by checking validity of transactions.

- Consensus layer refers to the different consensus algorithms used such as Proof-of-Work, Proof-of-Stake and others.

- Data layer contains various transaction data in blocks.

Figure 10 shows the illustration of Blockchain layers.

9 Appendix 3

9.1 Duplicated Transactions: Bitcoin Core source code

In this section, we will explore how the Merkle Tree Hash is vulnerable in some ways. As mentioned earlier, in the event of odd-numbered transactions being formed. The last transaction will be duplicated in the Merkle hash tree. The merkle root function can then be effectively preimaged by changing the input so that one of the intermediate lists is of even length with the last two elements equal. This duplication potentially leads to a scenario where the we have two Merkle trees: merkleHash([a, b, c, d, e, f]) == merkleHash([a, b, c, d, e, f, e, f]), a collision.
As a result, two blocks can easily be created that have the same block hash, though one can be valid and the other invalid, by duplicating one or more of the transactions in a way that maintains the Merkle root hash. Duplicating any

transaction will make the block invalid, since the block double spends a certain past transaction output.

Solution
Currently, this exploitation is marked as CVE-2012-2459, and majority of the Bitcoin network is already protected against this vulnerability. The fix for this exploitation was completed by Gavin Andresen by rejecting blocks with duplicated transaction in CheckBlock(), preventing them from being cached at all.

9.2 Bypassing Trans-ForgeFree

There exist attacks on blockchain where users in a network are isolated with no live view of the current ledger. As such, a malicious attacker will be able to perform malicious transactions with the user whilst still ensuring that the transactions are valid with the its digital signature. One such attack is the Eclipse attack.

Introduction to Eclipse Attack: An Eclipse Attack is a means of attacking a decentralized network through which an attacker seeks to isolate and attack a specific user. A successful Eclipse Attack enables a would-be bad actor to isolate and subsequently prevent their target from attaining a true picture of real network activity and the current ledger state. This attack is made possible because a decentralized network does not let all nodes simultaneously connect to all other nodes on the network. Instead, for efficiency, a node connects to a select group of other nodes, who in turn are connected to a select group of their own.

How it occurs: For the attack to occur, the malicious actor will ensure that all of the target's connections are made to attacker-controlled nodes. The entity will first flood the target with its own IP addresses, which the victim will likely connect to upon the restart of their software. A restart can either be forced (i.e. with a DDoS attack on the target), or the attacker can simply wait for it to occur. Once this has occurred, the unsuspecting victim is at the mercy of the malicious nodes – with no view of the wider network, they can be fed incorrect data by the attacker.

Bypassing of Trans-ForgeFree: Once a malicious actor has isolated a user by taking control of all outgoing connections from the user, he will be able to control the isolated user's view on the ledger. Hence, if User A is the malicious actor, User B is the isolated node and User C is another network entity, then User A would be able to send a payment to User C and then send the same transaction to User B. User B is unaware that those funds have already been spent as all his outbound connections route are through User A who is able to suppress and manipulate what information User B receives.

This shows that an attacker can still have a valid signature and yet create a malicious transaction that will be valid as well, bypassing the security of the digital signature. Hence, the attacker will still stay Trans-ForgeFree and create malicious transactions to his victims.
Attacks such as 0-confirmation double spends, N-confirmation double spends and selfish mining occurs due to the Eclipse attack.

9.3 51% attack:

This attack is carried out by a group of miners who are capable of controlling more than 50% of the network's computing power. They would be able to prevent new transactions from gaining confirmations, and halt payments between some or all users[13]. Although a 51% attack would not completely destroy the entire blockchain network, it still gives the attacker a possibility of reversing transactions that were completed, thereby violating our Time-Secure principle.

How does it work: While formulating an experiment for the Time-secure principle, we saw how a blockchain network functions. It will consist of a lot of nodes, each of which will be working hard to compute the proof-of-work for new blocks to be added to the chain. If a node receives conflicting blocks in the chain, it records both the versions and selects the chain of blocks that grows longer after a period of time. The 51% attack attacks this principle.
In our proof for Time-Secure, we saw that the probability of an attacker trying to alter blocks in the blockchain drops exponentially as the number of blocks the attacker has to catch up with increases. But if the attacker is given more than half of the computing power of the network, then the chances of the attacker succeeding can increase.

Likelihood of 51% attack: As a blockchain network grows bigger and bigger in size, it makes the chances of a 51% attack taking place less likely. This is because the cost of performing a 51% attack is proportional to the the amount of computational power attached to the network. Therefore, the bigger the network and the more nodes it contains, more hash power is needed to control over 50% of it.
But even if an attacker were to reach above 50% of the hashrate, the size of a blockchain could still provide security. Because blocks are linked together in the chain, a block can be altered only if all subsequently confirmed blocks are eliminated since each block would be containing the hash of the previous block[7].

Security Formulation in Contact Tracing Systems - A Dive Into Privacy and Integrity*

LI JIANHAN[1 [E0406503]],
NG CHOON WAH[2,3 [E0310972]],
OOI QIU JIA[3 [E0335622]],
YONG MING YANG[3 [E0406866]], and
ZHENG SHAOPENG[3 [E0322763]]

National University of Singapore

Abstract. This paper presents the security formulations in the implementation of contact tracing systems, and highlights the potential vulnerabilities in such systems. The privacy preserving and data integrity aspects are also examined through the lens of adversarial games, constructions and proofs.

Keywords: Contact tracing · Privacy · Integrity · Security formulation.

1 Introduction

As COVID-19's devastating impact spread across the world, governments sought to mitigate the damage via the use of bluetooth-enabled contact tracing applications. While the implementations tend to be varied, most contact tracing systems are based around similar architectures. By proposing adversarial games, constructions and proofs, we examine how certain constructions can be proven to be secure. The paper begins with a discussion on privacy preservation, where the notions of perfect privacy, standard differential privacy and ε-differential privacy are explored. Following which, we dive into the discussion on data integrity preservation, where notions of unforgeability and indistinguishablility comes into picture.

2 Privacy

Contact tracing systems consist of a database to store user particulars and close encounters with COVID-positive individuals. The ability to perform statistical queries on databases is essential, as it allows health authorities to draw insights into the current situation surrounding the pandemic. With each query, however, the risk of exposing an individual's data increases, resulting in accumulative privacy loss.

* Supported by National University of Singapore.

In the following sections, we will progressively develop a holistic definition of a "privacy-preserving" system, by not only considering the privacy aspect of a system but also the practicality of such a system given the limitations on privacy loss. Finally, we will construct a system which achieves a good compromise between privacy and utility.

2.1 Defining the System

Consider a general Contact Tracing System where besides being able to collect data from individuals, the system also allows data analysts to perform statistical analysis on the database. This allows health authorities to gain valuable insights, such as the number of cases in specific regions and the average age of those diagnosed with COVID.

The Contact Tracing System

- **DbInit:** initializes data set with records each containing a set of attributes $A = \{A_1, A_2, ..., A_n\}$.
- **Analyse:** queries the database and computes statistical data. Outputs the final result to the data analyst.

2.2 Perfect Privacy

Intuitively, a contact tracing system achieves perfect privacy if there is zero privacy loss from analysing the data set. Privacy loss here can be seen as a measure of how much new information an adversary can learn about any specific person X in the database by analysing the database which contains X.

DEFINITION 2.1 Let K_P denote the information known about any person P in data set D *before* analysis, and K'_P denote the information known about P *after* analysis. Then, a contact tracing system Π is perfectly private if $K_P = K'_P$.

Differential Privacy Approach An alternative approach to study privacy loss is through differential analysis. The main idea is to generate two different inputs or data sets D_0 and D_1, with a difference of one person and we compare the differences in the analysis outcomes to determine if privacy is compromised. An algorithm for analysing the two data sets is said to be differentially private if by examining the two outputs from the algorithm, one cannot differentiate whether an individual is part of D_0 or D_1.

Let Π denote the contact tracing system defined in Section 2.1, consider the following experiment Diff-Priv$_{\mathcal{A},\Pi}$ defined for Π = (DbInit, Analyse) and adversary $\mathcal{A}$:

The differential privacy preservation experiment Diff-Priv$_{\mathcal{A},\Pi}$:

1. Run DbInit() to initialize two identical data sets.

2. Challenger chooses a random bit $b \in \{0, 1\}$, and randomly removes a Person X from D_b, such that $D_b = D_{b^*} \setminus X$, where b^* is the bit-wise complement of b. Example: suppose $b = 1$, Person X will be removed from D_1, such that $D_1 = D_0 \setminus X$.
3. The adversary is given access to an Analysis Oracle M. Oracle M submits a pair of queries, (Q_b, Q_{b^*}), to D_b and D_{b^*}. After obtaining a pair of results (R_b, R_{b^*}), M aggregates the raw data into statistical data (M_b, M_{b^*}), before passing them to the adversary. Finally, the adversary outputs $b' = 1$ if it believes that Person X was removed from D_1, and 0 otherwise.
4. The output of the experiment is 1 if and only if $b = b'$, and 0 otherwise.

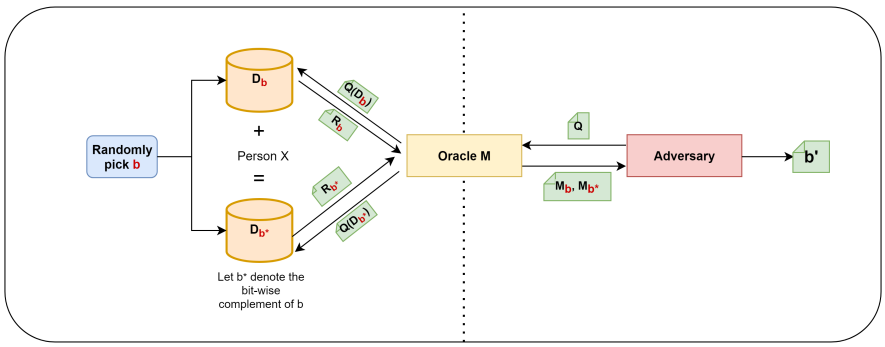


Fig. 1. The Differential Privacy Preservation Experiment

DEFINITION 2.2 A Contact Tracing System Π is perfectly private if for all adversary $\mathcal{A}$,

$$Pr[\text{Diff-Priv}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$$

Proof. We want to prove that if a system Π is differentially private, then Π is perfectly private.

Consider a data set with N people. If we let P_i be the i^{th} person in the data set, then based on our initial assumption that $\Pr[\text{Diff-Priv}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$, the adversary should not be able to tell apart a data set which includes P_i and a data set which excludes P_i with probability more than 1/2. This implies that the adversary is not able to uncover any new information about P_i, because otherwise, the adversary would have successfully deciphered the data set which contains P_i with a probability more than 1/2, given the new information uncovered about P_i, contradicting our assumption that $\Pr[\text{Diff-Priv}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$.

Thus, we conclude that for *any* P_i where $i \in \{1, 2, ..., N\}$, $K_{P_i} = K_{P_i'}$. Therefore, a contact tracing system Π that is differentially private with no privacy loss, i,e. $Pr[\text{Diff-Priv}_{\mathcal{A},\Pi} = 1] = \frac{1}{2}$, is also perfectly private.

So far we have discussed the definition of perfect privacy and shown that differential privacy can be used to determine if a system is perfectly private. In addition, we also want to answer the question of whether a perfectly private system can exist, and if so, whether it is practical for use in contact tracing systems.

Consider a more specific implementation of a contact tracing system $\widetilde{\Pi}$, where the set of attributes in the data set consists of basic contact tracing and user information such as the user's temporary ID, age and location. The analysis algorithm queries and outputs the mean age of all individuals in the database and returns it.

CONSTRUCTION 2.1

- **DbInit():** initializes data set with records each containing a set of attributes $A = \{A_1, A_2, ..., A_n\}$. The data for each attribute is evenly distributed about the mean.
- **Analyse():** queries the database and computes the statistical mean $R \leftarrow \frac{1}{n}\sum_{i=1}^{n} a_i$ where a_i is the value of the attribute a for row i.

CLAIM 2.1 Let $\widetilde{\Pi}$ be a Contact Tracing System implemented based on Construction 2.1. Assuming that an adversary has no prior knowledge of any person in the data set, and age is a positive integer, $\widetilde{\Pi}$ is a perfectly private Contact Tracing System.

Proof. Let $\mathcal{A}$ be an adversary who has zero knowledge of anyone in the data set. We first proof that $\widetilde{\Pi}$ is differentially private, which would also imply that $\widetilde{\Pi}$ is perfectly private via Definition 2.2.

Suppose we have two identical data sets D_0 and D_1 where D_1 does not include Person X, such that $D_1 = D_0 \setminus X$. Given the mean ages, M_0 and M_1, for data sets D_0 and D_1, there are two cases to consider:

1. $M_0 < M_1$: we further break it down into two possibilities (a) and (b):
   (a) Person X $\in D_0$, or equivalently $\text{Age}_X < M_1$ (We denote Person X's age with Age_X).
   $Pr[X \in D_0] = Pr[Age_X < M_1] = \frac{1}{2}$ since we assume that ages are evenly distributed about the mean based on Construction 2.1. In other words, the number of people with age smaller than the mean is equal to the number of people with age larger than the mean, and thus it is equally likely to pick a person X with age greater or smaller than the mean.
   (b) Person X $\in D_1$, or equivalently $\text{Age}_X > M_0$.
   $Pr[X \in D_1] = Pr[Age_X > M_0] = \frac{1}{2}$ since we assume that ages are evenly distributed about the mean based on Construction 2.1.
2. $M_0 > M_1$: same argument as Case 1, except that we calculate the probabilities $Pr[X \in D_0] = Pr[Age_X > M_1] = \frac{1}{2}$ and $Pr[X \in D_1] = Pr[Age_X < M_0] = \frac{1}{2}$.

By exhausting all the possible cases above, we reach a conclusion that $Pr[X \in D_0] = Pr[X \in D_1]$. Linking back to the Diff-Priv$_{\mathcal{A},\widetilde{\Pi}}$ experiment in Section 2.2, the probability that an adversary $\mathcal{A}$ will guess correctly which data set includes or excludes Person X is thus exactly 1/2,

$$Pr[\text{Diff-Priv}_{\mathcal{A},\widetilde{\Pi}} = 1] = \frac{1}{2}$$

Therefore, the system $\widetilde{\Pi}$ is differentially private with *zero privacy loss*, which also implies that $\widetilde{\Pi}$ is perfectly private by Definition 2.2.

2.3 Limitations of Perfect Privacy / Privacy-Utility Trade-off

As we have seen, perfect privacy is achievable, by carefully limiting the queries and ensuring that data has an even distribution. However, one thing you may have also noticed is that such a system, despite being perfectly private, is not very useful. For instance, health authorities who wish to know more about the current pandemic will be limited by how much analysis can be done on the database. The more limitation being placed on the queries allowed to be performed on the data set, the less utility and insights we gain from the data set. Moreover, it may not always be possible to have even distribution of data in our database. For example, there might be a disproportionate number of people in a certain age range, which may allow the adversary to guess which data set Person X is from with a probability greater than 1/2.

Therefore, even if perfect privacy is attainable, it comes with a significant trade-off. Depending on the severity of the pandemic, being less restrictive can allow health authorities to take pre-emptive measures, potentially saving thousands of lives.

2.4 A Weaker Notion of Privacy

The glaring trade-off of perfect privacy begs the question: *is it possible to relax perfect privacy to increase utility and yet, does not compromise too much on the privacy of the population?* In this section, we define a weaker notion of privacy by introducing an additional parameter, ε, into our privacy preservation definition.

While more privacy is always desirable, it may be acceptable to allow some privacy loss as long as it does not exceed a set threshold, in exchange for greater utility. Therefore, we relax our definition of "privacy-preserving" to accept an acceptable amount of privacy loss bounded by ε. The experiment from Section 2.2 remains unchanged, but the security property is revised:

DEFINITION 2.3 (Dwork, 2014) A system is ε-differentially private if for all $\mathcal{PPT}$ adversary $\mathcal{A}$, there is a ε, s.t.

$$\frac{Pr[M_b \in O]}{Pr[M_{b^*} \in O]} \leq e^{\varepsilon}$$

where O refers to the output space of M, M_b and M_{b^*} are the statistical data computed on data sets D_b and D_{b^*}.

To achieve ε-differential privacy, one could introduce a noise generation algorithm to control the amount of privacy loss. By adding suitable amount of noise, we can minimise privacy loss while maintaining the accuracy of the information. On the other hand, introducing too much noise may be an over-kill, resulting in information loss and poor utility. An example of such a noise generation algorithm is as follows:

ALGORITHM 2.1 For every row of data retrieved from the database, we perform the following on each of the attributes:

1. Uniformly choose a bit, $b \in \{0, 1\}$.
2. If $b = 1$, return the result without modification.
3. If $b = 0$, partition the set of all possible values for the attribute, A, into two sets A_0 and A_1 of equal size, choose randomly a second bit $b' \in \{0, 1\}$ and uniformly select a value from set A_1 if $b' = 1$ and from set A_0 if $b' = 0$.

To demonstrate the idea of bounded privacy loss, we construct a system which will be shown to be ε-differentially private, with ε privacy loss.

CONSTRUCTION 2.2

- **DbInit():** as before
- **Analyse():** obtain a set of data R from the database, adds noise to the data by computing N(R), where N is Algorithm 2.1. On input N(R), compute and return the statistical data.

CLAIM 2.2 (For simplicity, we assume the data set contains only boolean attributes) A Contact Tracing System Π based on Construction 2.2 is (ln 3)-differentially private.

Proof. Given an attribute A, as according to Algorithm 2.1, we partition the set of all possible values into two equal size sets, $A_0 = \{\text{True}\}$ and $A_1 = \{\text{False}\}$. We compute the conditional probability that the output of the noise generation algorithm is True given that the actual value of attribute A is True.

$$Pr[\text{ Outcome} = \text{True} \mid \text{Actual} = \text{True }] = \tfrac{3}{4}$$

Specifically, the probability of outcome to be True given that the actual is True $= \Pr[b = 1] + \Pr[b = 0 \text{ and } b' = 0] = \frac{1}{2} + \frac{1}{4} = \frac{3}{4}$.

Whereas the probability of the outcome to be True given that actual is False is given by:

$$Pr[\text{ Outcome} = \text{True} \mid \text{Actual} = \text{False }] = \tfrac{1}{4}$$

where $b = 0$ and $b' = 0$.

We apply the same logic to the False case to arrive at the following:

$$\frac{Pr[\text{ Outcome = True } \mid \text{ Actual = True }]}{Pr[\text{ Outcome = True } \mid \text{ Actual = False }]} = \frac{3/4}{1/4} = 3$$

and

$$\frac{Pr[\text{ Outcome = False } \mid \text{ Actual = False }]}{Pr[\text{ Outcome = False } \mid \text{ Actual = True }]} = \frac{3/4}{1/4} = 3$$

Hence, Construction 2.2 is (ln 3)-differentially private by Definition 2.3, where $e^{\varepsilon} = 3$ and thus $\varepsilon =$ln 3.

In conclusion, ε-differentially private system guarantees that for any $\mathcal{PPT}$ adversary $\mathcal{A}$, the ratio of the probability of observing outcome M_b and M_{b*} is within e^{ε}, such that observing these two outputs makes it infeasible for the adversary to identify which data set includes the individual.

3 Integrity

In this section, we will first discuss the notion of data integrity in the context of contact-tracing, before moving on to the formal definition of the security property using adversarial games and a construction.

3.1 Contact Tracing Applications

Context of Contact Tracing Applications When it comes to Integrity, we will be looking at parts of applications implementing the BlueTrace protocol. These parts include temporary IDs (**tempIDs**) and **encounter messages**. As contact tracing applications have their own varying implementation, temporary ID will be known as ID to avoid confusion.

IDs are generated on a backend server of the governing authority. The governing authority generates the ID by encrypting the assigned userID, start time and expiry time, and subsequently appending an IV and Authentication Tag. IDs have short lifetimes which are determined by the governing authority, after which, they expire and the next ID is used. This notion shall be known as **refreshing** of IDs. They are retrieved in batches via the mobile application and are stored locally on the mobile device.

3.2 Data Integrity

Data Integrity will be defined as such:

When an adversary is able to spoof a specific user even after the ID of the user has been **refreshed**, and subsequently send compromised data which is deemed valid by the server.

In this section we will be exploring the following questions: How can we ensure that only authentic IDs are accepted into the database? What is the probability of an adversary successfully creating a valid ID that passes the verification?

The integrity of the database is said to be compromised if the database is unable to identify whether the IDs in the close contact log were sent by the adversary or the actual user.

3.3 Scheme

Before we formally define the integrity of IDs, we first construct a scheme for the ID and define how it is used. Two parties, A and B, attempts to communicate in an authenticated manner. Party A produces a set of IDs and shares it with B in advance of the communication. When B communicates with A later on, B sends the message along with one of the IDs shared with B earlier on. Party A can then verify the authenticity of the message by verifying the ID attached to it. Formally,

DEFINITION 3.1 An ID scheme consists of three probabilistic polynomial-time algorithms (Gen, Create, Verify) such that:

- Gen(1^n): Key-generation algorithm that takes in as input a unique identifier (a security parameter of length n) and outputs a key k with $|k| < n$.
- Create$_k$(): ID-generation algorithm that takes in as input a key k and outputs an ID.
- Verify$_k$(id): Verification algorithm that takes in as input a key k and an id. It outputs 1 when the id is valid, and 0 otherwise.

It is noted that Gen and Verify are both deterministic, while Create is a probabilistic algorithm.

3.4 Unforgeability

Context of Unforgeability As contact-tracing relies heavily on IDs encapsulated in encounter messages and exchanged via bluetooth, it is of utmost importance that these IDs cannot be forged. Consider the scenario whereby an adversary is able to forge the IDs of a user. The adversary would then be able to log fake encounter messages with other users, and thereby spreading false information when the encounter messages history are reviewed by the governing authority. This therefore affects the integrity of the data used for contact tracing purposes. Hence, we analyze the scenario through the ID authentication experiment explained below.

The ID authentication experiment, IDForge$_{A,\Pi}$

1. On input of length n, Challenger use Gen to create a key k.
2. Adversary has access to an Oracle, which uses Create$_k$ to reply to a query.

3. Q contains all the replies the Oracle has made to the adversary.
4. Adversary outputs an ID and passes it to the Challenger.
5. Challenger outputs 1 if the ID is not in Q, and is valid using Verify_k, 0 otherwise.

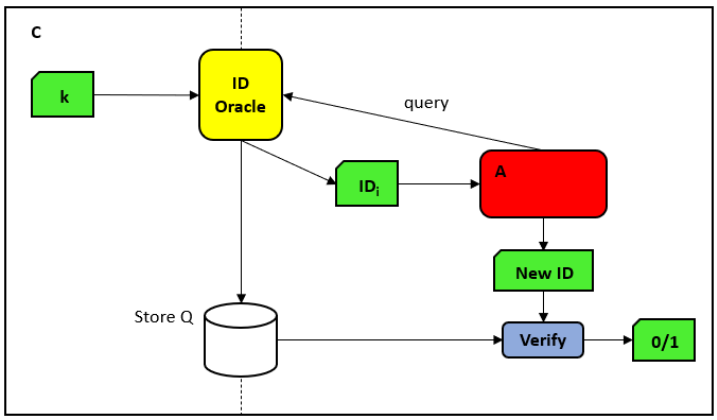


Fig. 2. $\text{IDForge}_{A,\sqcap}$ experiment

DEFINITION 3.2 A scheme $\sqcap$ preserves unforgeability if, for any Adversary $\mathcal{A}$, there is a negl, s.t.

$$Pr[\text{IDForge}_{\mathcal{A},\sqcap}(n) = 1] \leq negl(n)$$

3.5 Distinguishability

Context of Distinguishability Assuming that the adversary is able to obtain all the IDs (updated every refresh) in the neighbourhood. If the adversary is able to distinguish between the refreshed ID among all IDs with larger than negligible probability, then the adversary can consistently pretend to be a specific user. Thus, this compromises data integrity. Hence, we analyze the scenario through the ID distinguishing experiment explained below.

Given two IDs, $(\text{ID}_i, \text{ID}_j)$ which are generated from (k_0, k_1) respectively by the oracle, we want to check if the adversary can distinguish which ID (i or j) will be refer to the same identity as ID_b after it expired.

The ID Indistinguishability experiment, $\text{IDIndistinguishability}_{A,\sqcap}$

1. Given two different keys k_0 and k_1, the Challenger will generate ID_0 and ID_1 respectively.

2. Let b be a randomly chosen binary value, and ID_b will be sent to the Adversary.
3. Adversary will query both ID-Oracles to output ID_i and ID_j, where ID_i will be the replacement ID for ID_1, and ID_j will be the replacement ID for ID_0.
4. Given ID_i and ID_j, adversary will have to choose which of them is the replace ID for ID_b.
5. Adversary outputs b' = 0 if ID_j is chosen, b' = 1 if ID_i is chosen.
6. Challenger outputs 1 if b = b', 0 otherwise.

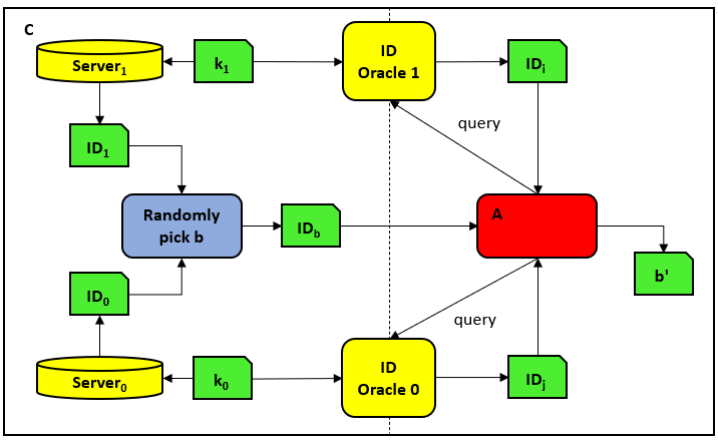


Fig. 3. IDIndistinguishability$_{\mathcal{A},\sqcap}$ experiment

DEFINITION 3.3 A scheme $\sqcap$ preserves indistinguishability if, for any Adversary A, there is a negl, s.t.

$$Pr[\text{IDIndistinguishability}_{A,\sqcap}(n) = 1] \leq \frac{1}{2} + negl(n)$$

Note that Experiment 2 is much more stringent property than Experiment 1. The definition implies that adversary cannot even distinguish the new ID from other IDs.

3.6 Construction

In this section, we consider a specific implementation of ID in Construction 3.1 using Definition 3.1 as shown below.

CONSTRUCTION 3.1

Implementation of ID Given a pseudorandom function F and a collision resistant cryptographic hash function $\mathcal{H}$, the server generates and verifies ID as follows:

- $\text{Gen}(1^n)$: on input a of length n, output $k = \mathcal{H}(a)$, where a is the NRIC of a user (or any unique identifier).
- $\text{Create}_k()$: on input k and uniform r, output $\langle F_k(r), \mathcal{H}(F_k(r))\rangle$
- $\text{Verify}_k(\text{id})$: on input $id = \langle m, t\rangle$ and k, output 1 if $t = \mathcal{H}(m)$, 0 otherwise.

3.7 Proofs

In this section, we will provide claims and proofs of our construction with respect to the security properties defined in Section 3.4 and 3.5

CLAIM 3.1 Let Π be an ID generation system based on Construction 3.1. Assuming that an adversary has access to the expired IDs of a certain user, Π preserves unforgeability.

Proof. In Construction 3.1, $\mathcal{H}$ is a collision resistant cryptographic hash function. This implies that it should be computationally infeasible for any adversary to find a different x' such that $\mathcal{H}(x') = \mathcal{H}(x)$. The adversary should not be able to forge an ID-tag pair with a probability better than $negl(n)$. Hence,

$$Pr[\text{IDForge}_{A,\Pi}(n) = 1] \leq negl(n) + negl(n) = negl(n)$$

CLAIM 3.2 Let Π be an ID generation system based on Construction 3.1. Assuming that an adversary has access to the expired IDs of a certain user, Π preserves indistinguishablilty.

Proof. Since r is uniformly chosen for every ID and F is a pseudorandom function in Construction 3.1, the output should still appear as though it was a random string to the adversary, even when the adversary has access to the past IDs. With these properties of F and $\mathcal{H}$, it should be computationally infeasible for any adversary to differentiate who the refreshed ID refers to. The probability of the adversary guessing the answer correctly is no better than a random guess ($\frac{1}{2}$) with $negl(n)$. Hence,

$$Pr[\text{IDIndistinguishability}_{A,\Pi}(n) = 1] \leq \frac{1}{2} + negl(n)$$

4 Conclusion

In conclusion, this paper focused on two key aspects of the contact tracing application; privacy and integrity. In the discussion on privacy, the paper motivates the notion of perfect privacy, the limitations that come with it, as well as a weaker notion of privacy. On the other hand, the discussion on integrity leads us to investigate the notions of unforgeability and indistinguishability, as well as construction and proofs to show that the mentioned security properties hold. Having explored both of these aspects, we now have a better understanding of contact tracing applications.

In reality, applications such as TraceTogether implemented by Singapore make use of encounter message and encounter message history for contact tracing. Encounter messages are basically a wrapper that includes the temporary ID, and a few other details, while encounter messages history refer to the list of encounter messages exchanged when two devices are in close contact. Although it is generally a robust system, we found that the integrity of the application was also highly dependent on factors such as expiry time for the temporary IDs, and methods of exchanging encounter messages. For example, Bluetooth technology is known to be prone to the Bluetooth Impersonation AttackS (BIAS). This allows for encounter messages to be intercepted. In implementations where the expiry time is unnecessarily long, adversaries are able to abuse the vulnerability and spread false encounter messages. Hence, although the general design of contact tracing applications are privacy and integrity preserving, emphasis should be placed on the details of the implementations.

References

1. Alvin, T., Chai, S.H., Janice, T., Jason B., Joel, K., Lai, Y., & Tang A.Q. BlueTrace: A privacy-preserving protocol for community-driven contact tracing across borders [White paper]. Government Technology Agency, Singapore. https://bluetrace.io/static/bluetrace_whitepaper-938063656596c104632def383eb33b3c.pdf
2. Dwork.C, Roth.A. (n.d.). The Algorithmic Foundations of Differential Privacy. Retrieved November 6, 2021, from https://www.cis.upenn.edu/ aaroth/Papers/privacybook.pdf
3. Differential Privacy. (n.d.). Harvard University Privacy Tools Project. Retrieved November 6, 2021, from https://privacytools.seas.harvard.edu/differential-privacy
4. Tyagi, N. (2017, May 17). What is Differential Privacy and How does it Work? — Analytics Steps. Retrieved November 6, 2021, from https://www.analyticssteps.com/blogs/what-differential-privacy-and-how-does-it-work
5. Differential Privacy for Privacy-Preserving Data Analysis: An Introduction to our Blog Series. (2020, August 4). NIST. Retrieved November 6, 2021, from https://www.nist.gov/blogs/cybersecurity-insights/differential-privacy-privacy-preserving-data-analysis-introduction-our
6. Nguyen, A. (2021, July 31). Understanding Differential Privacy - Towards Data Science. Medium. Retrieved November 6, 2021, from https://towardsdatascience.com/understanding-differential-privacy-85ce191e198a