Modern Cryptography: Lecture 14

Selected Topics

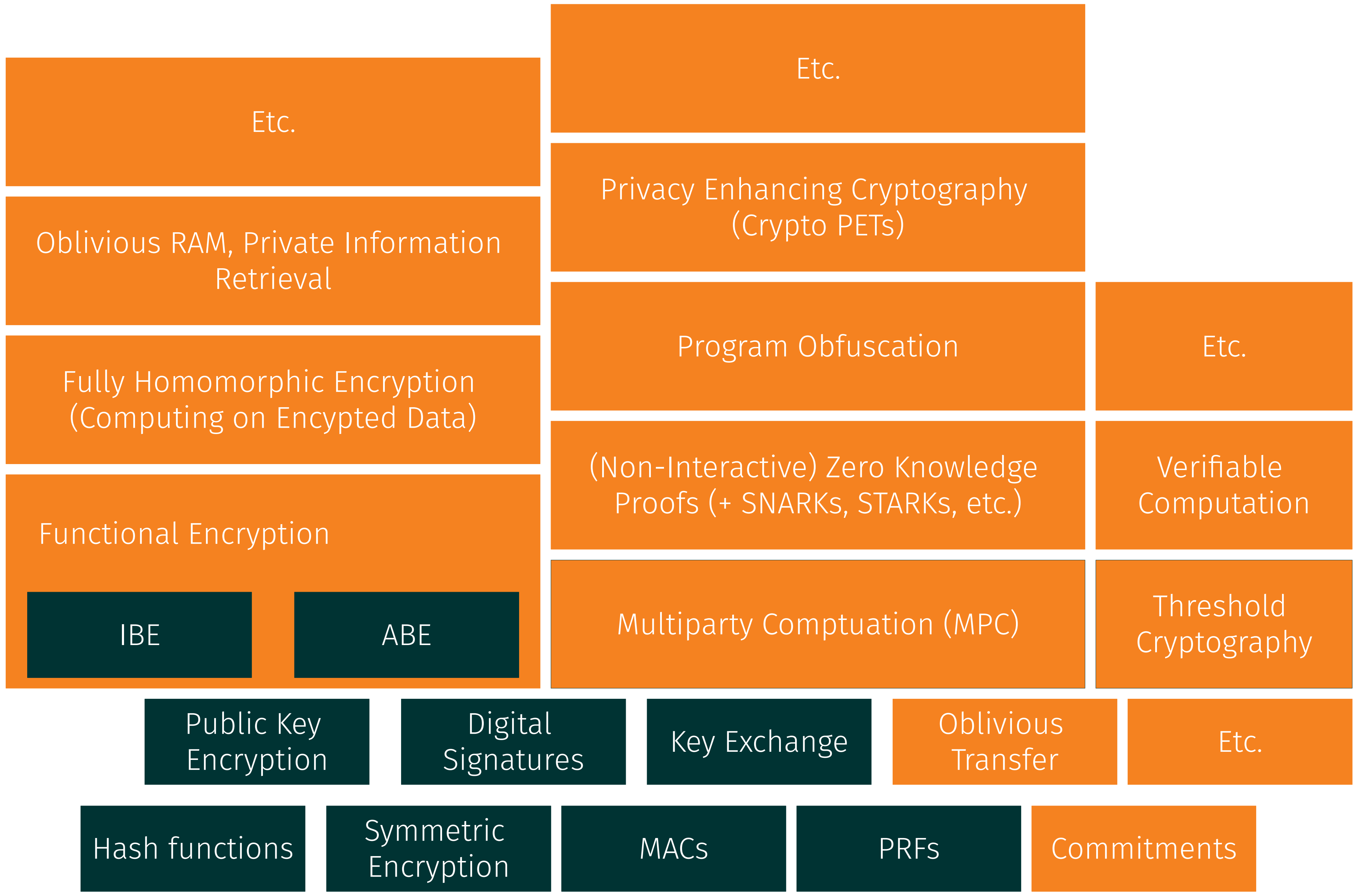
Daniel Slamanig

AUSTRIAN INSTITUTE OF TECHNOLOGY

Organizational

- Where to find the slides and homework?
  - https://danielslamanig.info/ModernCrypto19
- How to contact me?
  - daniel.slamanig@ait.ac.at
- Tutors: Guillermo Perez, Karen Klein
  - guillermo.pascualperez@ist.ac.at; karen.klein@ist.ac.at
- Official page at TU, Location etc.
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=3463&dsrid=417&courseNr=192062&semester=2019W
- Tutorial, TU site
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=3593&dsrid=246&courseNr=192063
- Exam for the second part: Thursday 30.01.2020 15:00-17:00 (Tutorial slot)

Topics in Advanced Cryptography...


Etc.
Etc.
Privacy Enhancing Cryptography (Crypto PETs)
Oblivious RAM, Private Information Retrieval
Program Obfuscation
Etc.
Fully Homomorphic Encryption (Computing on Encypted Data)
(Non-Interactive) Zero Knowledge Proofs (+ SNARKs, STARKs, etc.)
Verifiable Computation
Functional Encryption
IBE
ABE
Multiparty Comptuation (MPC)
Threshold Cryptography
Public Key Encryption
Digital Signatures
Key Exchange
Oblivious Transfer
Etc.
Hash functions
Symmetric Encryption
MACs
PRFs
Commitments

Selected Topics

- Threshold cryptography
  - Distribute operations with the secret key sk among a set of parties
  - A certain number of participants need to be involved to perform an operation
- Brief primer on Multiparty Computation (MPC)
  - A number of parties can compute any function jointly without revealing their inputs to the other parties
- Puncturable Encryption
  - Public key encryption with “update capabilities” on the secret key
  - Secret key can be punctured on ciphertext s.t. this ciphertext can no longer be decrypted

Threshold Cryptography: Motivation

- If the secret key (of an encryption or signature scheme) is in a single location, this represents a single point of failure
  - Problem that happened in practice, e.g., with Bitcoin ECDSA private keys
- We may want to enforce that a signature generation or decryption is only possible when a certain set of participants agree to do so
- Idea
  - Let a set of parties jointly generate a secret key ("shares" of the key may also be distributed to the parties by a trusted dealer)
  - The public key typically looks like a public key of the underlying scheme
    - So public key operations are as usual
  - Using the secret key (i.e., signing or decryption) requires an interactive protocol between (a subset of the) participants

Secret Sharing

- A dealer shares a secret key between n participants
- Each participant i ∈ {1,..., n} receives a share
- Predefined groups of participants (so called authorized groups) can cooperate to reconstruct the secret from their shares
- Unauthorized groups cannot get any information about the secret

- We will look at (k, n)-threshold secret sharing schemes
  - Every subset of at least k participants of the n participants can reconstruct the secret (is authorized)
  - Any subset of k–1 participants will get no information about the secret (is unauthorized)

(n,n)-Treshold Secret Sharing

- Let s be a secret from a finite group (G, +)
- The dealer chooses n-1 uniformly random elements $s_1, \dots, s_{n-1}$ from G and computes $s_n = s - (s_1 + \dots + s_{n-1})$
- The shares are $(s_1, \dots, s_n)$ and party i is given share s_i
- Given $(s_1, \dots, s_n)$, one can successfully recover $s = s_1 + \dots + s_n$
- Given s_i for $i \neq j$: $\Sigma_{i \neq j}\, s_i = s - s_j$ is uniformly random (no information)

- **Not robust at all!**
  - If a single participant fails to provide the share, reconstruction is not possible
- We are interseted in **(k,n)-threshold schemes** where **k<n**

Shamir Secret Sharing

- Basis
  - Given k points on the plane $(x_1, y_1), \dots, (x_k, y_k)$, all x_i distinct, there exists an unique polynomial f of degree $\leq k - 1$, s.t. $f(x_i) = y_i$ for all i
  - Holds also in the field Z_p for p prime
  - Constructive proof: Use Lagrange interpolation
- How is this used?
  - Let s be a secret in Z_p and the threshold be k
  - Dealer selects a random degree k-1 polynomial $f(x) = a_{k-1}x^{k-1} + \dots + a_1x + a_0$
    - Select $a_{k-1}, \dots, a_1$ uniformly random from Z_p and set $a_0 = s$
  - For $i \in \{1, \dots, n\}$, give the share $s_i = (i, f(i))$ to the i^{th} participant

Shamir Secret Sharing

- **Correctness:** the secret s can be reconstucted from every subset of k shares
  - Proof: By the Langrange formula, given k points (x_i, y_i), for i = 1, … ,k

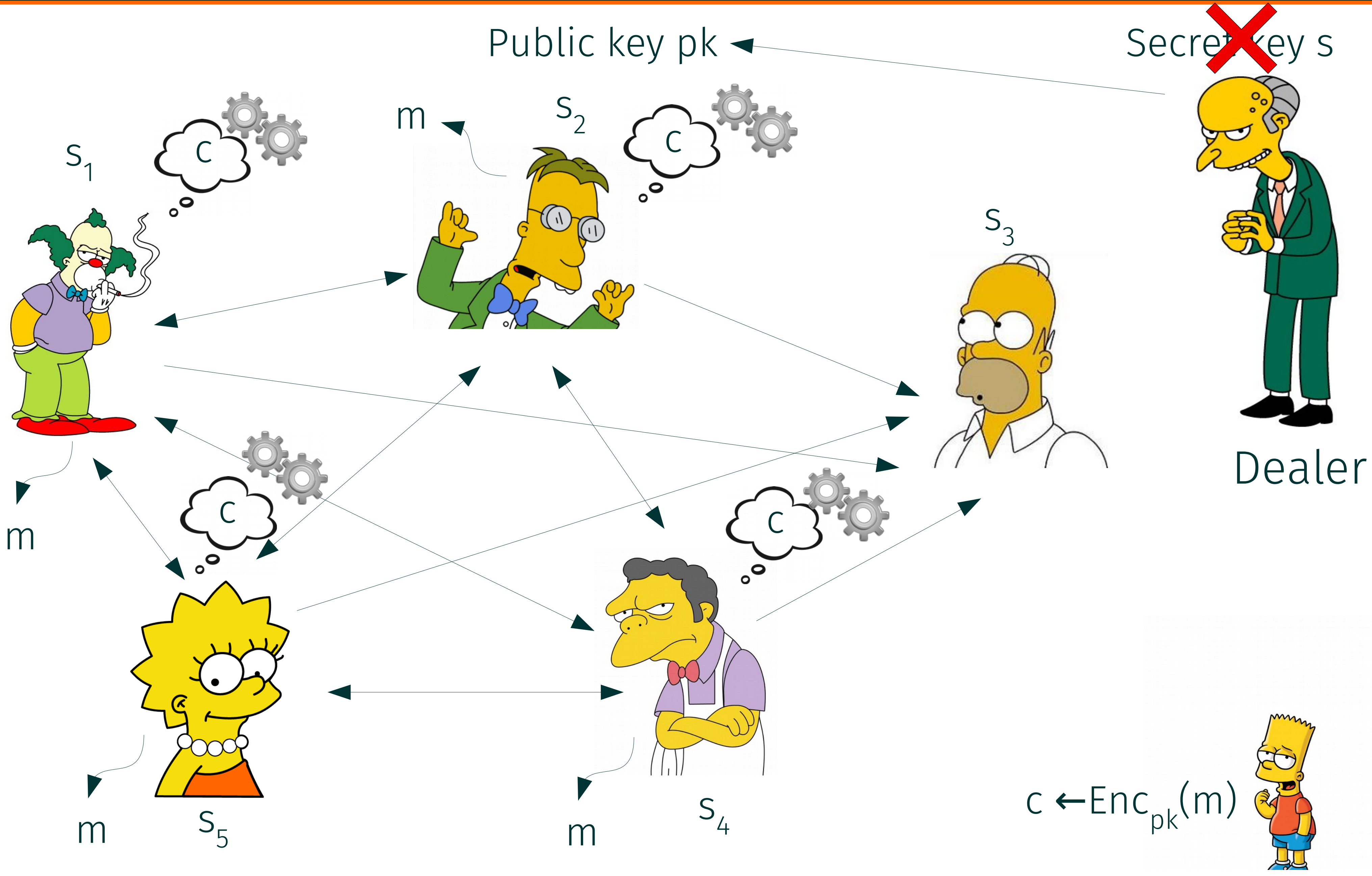
$$f(x) = \sum_{i=1}^{k} y_i \prod_{j=1, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \bmod p$$

  - and consequently

$$s := f(0) = \sum_{i=1}^{k} y_i \underbrace{\prod_{j=1, j\neq i}^{k} \frac{-x_j}{x_i - x_j}}_{\Delta_i} \bmod p.$$

- **Secrecy (perfect):** Any subset of up to k – 1 shares does not leak any information on the secret
  - Proof: Given k – 1 shares (x_i, y_i) every candidate secret $s' \in Z_p$ corresponds to a unique polynomial of degree k-1 for which f(0)=s'. For all $s' \in Z_p$ the probabilities Pr[s' = s] are equal.

Threshold Encryption



Threshold ElGamal Encryption

- Let z be the ElGamal secret key and $h := g^z$ the public key (we work in a group G of prime order q generated by g).
- Every participant i receives a share $s_i = (i, y_i)$ of z obtained from Shamir (k,n)-threshold secret sharing
- We observe (notice that Δ_j are publicly computable) that

$$z = \sum_{j \in X} y_j \Delta_j \quad \text{and} \quad g^z = \prod_{j \in X} (g^{y_j})^{\Delta_j}$$

- Given an ElGamal ciphertext $(c_1, c_2) = (g^r, mh^r)$ we assume a honst set X of t participants
  - Every participant j in X broadcasts $w_j := (c_1)^{y_j}$
  - Everyone in X can recover the plaintext as $m = \frac{c_2}{\prod_{j \in X} w_j^{\Delta_j}}$

Correctness: $\frac{c_2}{\prod_{j \in X} w_j^{\Delta_j}} = \frac{m(g^z)^r}{g^{r \sum_{j \in X} y_j \Delta_j}} = \frac{mg^{rz}}{g^{rz}}$

Threshold Cryptography: Remarks

- We have assumed that the parties participating in the decryption are honest
  - Malicious parties can enforce an incorrect result by publishing a malformed w_j value
  - Can be prevented by forcing the parties to prove that the w_j values are well formed (i.e., by attaching a non-interactive zero-knowledge proof)
- Can come up with threshold versions of various signature schemes
  - Schnorr, (EC)DSA, etc.
  - Somewhat hot topic today (cryptocurrencies)

Why you need threshold signatures to protect your wallet

Arvind Narayanan

Princeton

Joint work with

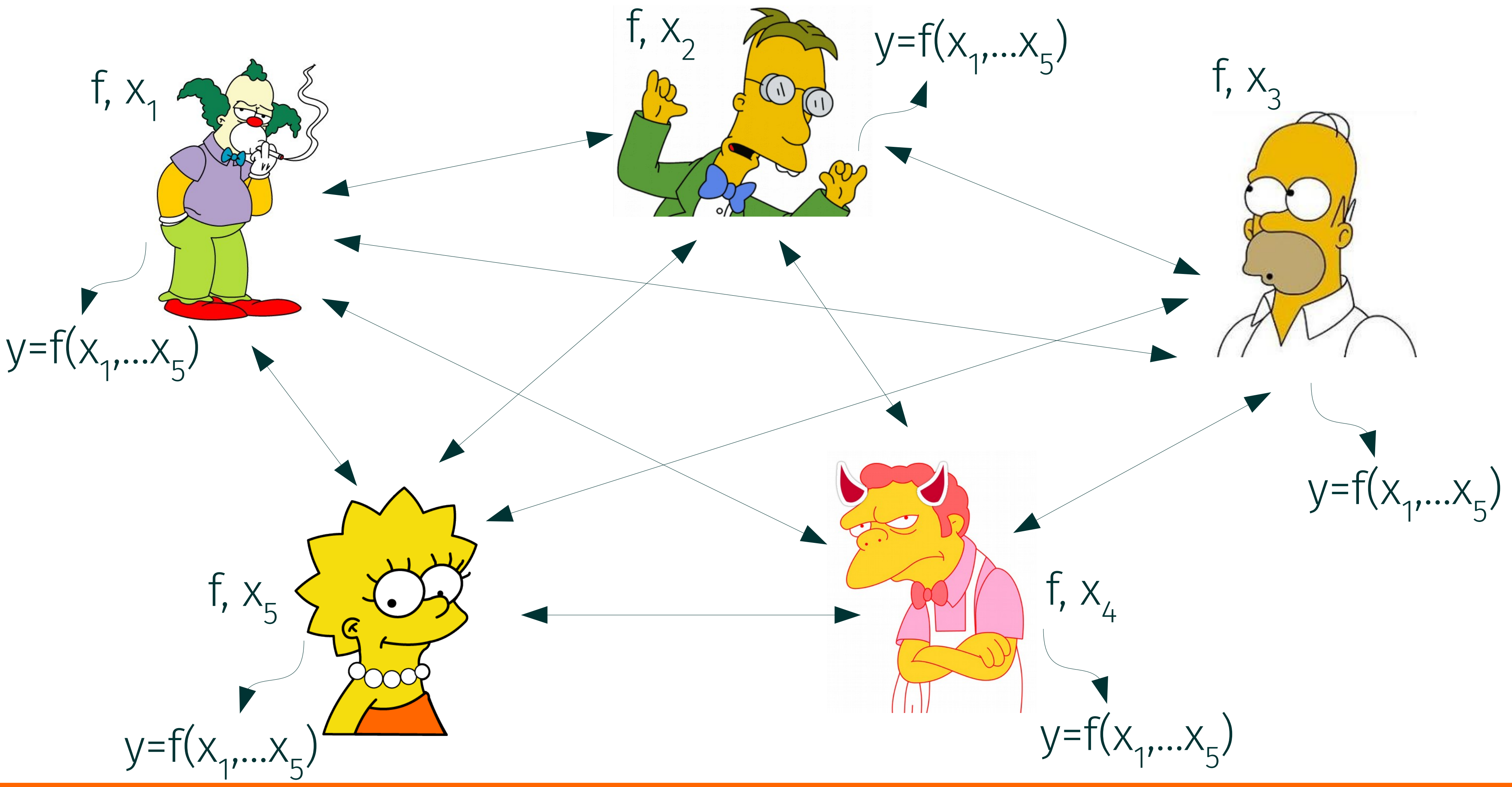
Steven Goldfeder · Harry Kalodner · Rosario Gennaro

Threshold Signatures are a 'Significant Milestone' in Bitcoin Security

430 Total views - Total shares

Multiparty Computation

- We have seen a very specific functionality computed in a distributed way without requiring the participants to reveal their secret inputs
- Can we do every computation in such a threshold manner? Yes!



Multiparty Computation

- We look at Ben-Or, Goldwasser, Wigderson (BGW) in a finite field $\mathbb{Z}_p$
  - Every possible function in $\mathbb{Z}_p$ is a polynomial
  - We need to show how we can do addition and multiplication

- BGW is a general MPC protocol that provides information theoretic guarantees
  - in the presence of semi-honest adversaries controlling a minority of parties ($< n/2$)
  - in the presence of malicious adversaries controlling less than a third of the parties ($< n/3$).

Multiparty Computation

- Use Shamir's (k,n)-threshold secret sharing with $k > n/2$ (honest majority)
- Every party i has a secret s_i and polynomial $f_i(0) = s_i$
- Every party j holds shares $f_i(j)$, $i \neq j$,

- **Addition:** Given $f_1(j)$ and $f_2(j)$ just add the shares: participants then share the polynomial $f_1 + f_2$ with $(f_1 + f_2)(0) = s_1 + s_2$.
- **Multiplication:** if $h = (f_1 \cdot f_2)$ then $h(0) = s_1 \cdot s_2$
  - However, h would have degree $\deg f_1 + \deg f_2 = 2k - 2$
  - Coefficients of h are not uniformly random
  - After every multiplication the parties perform a simple protocol that reduces the degree of h and adds uniformly random values to all coefficients of h, except to h_0

Multiparty Computation

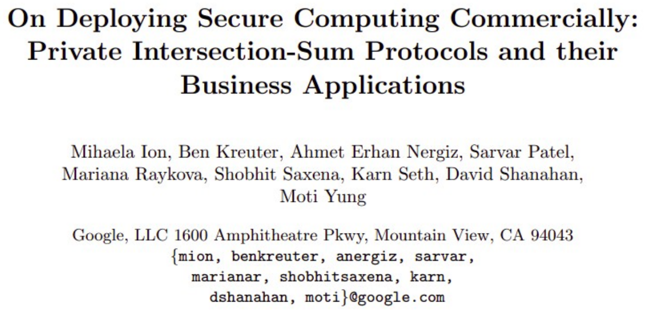
- Very hot and active topic nowadays

News · Technology

Multi-party Computation Alliance founded for digital privacy, security

2 months ago · by Miranda Wood

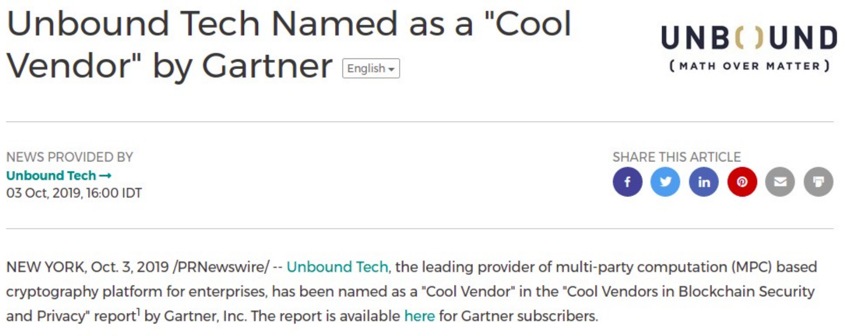
Today, several tech firms have become founding members of the Multi-party Computation (MPC) Alliance for security and privacy in the digital age. Members include developers and practitioners of MPC hoping to protect the increasingly valuable 'data footprint'.



On Deploying Secure Computing Commercially: Private Intersection-Sum Protocols and their Business Applications

Mihaela Ion, Ben Kreuter, Ahmet Erhan Nergiz, Sarvar Patel, Mariana Raykova, Shobhit Saxena, Karn Seth, David Shanahan, Moti Yung

Google, LLC 1600 Amphitheatre Pkwy, Mountain View, CA 94043
`{mion, benkreuter, anergiz, sarvar, marianar, shobhitsaxena, karn, dshanahan, moti}@google.com`



Unbound Tech Named as a "Cool Vendor" by Gartner English ▾

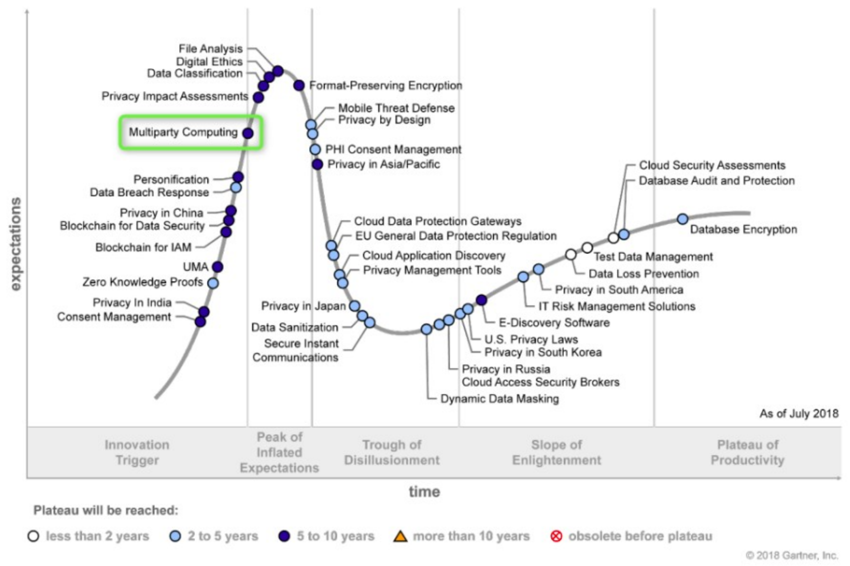
UNB( )UND (MATH OVER MATTER)

NEWS PROVIDED BY
Unbound Tech →
03 Oct, 2019, 16:00 IDT

SHARE THIS ARTICLE

NEW YORK, Oct. 3, 2019 /PRNewswire/ -- Unbound Tech, the leading provider of multi-party computation (MPC) based cryptography platform for enterprises, has been named as a "Cool Vendor" in the "Cool Vendors in Blockchain Security and Privacy" report[1] by Gartner, Inc. The report is available here for Gartner subscribers.

Figure 1. Hype Cycle for Privacy, 2018



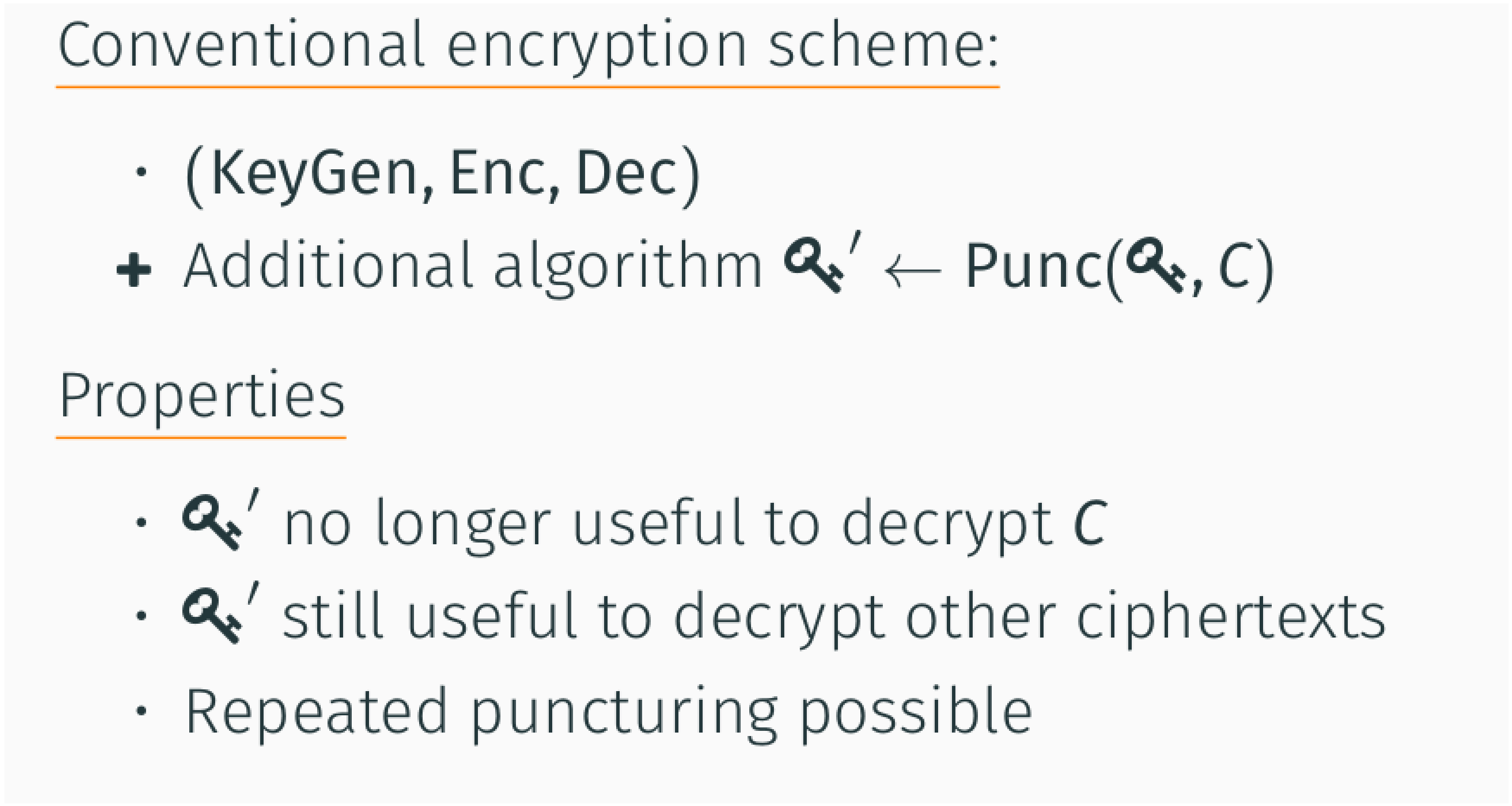
How to do big data without being creepy

We need ways to use big data sets in ways that don't compromise individual privacy, especially when data is shared between multiple organizations. Here's a look at some promising approaches and tools

By Gordon Haff | January 16, 2020 — 8 readers like this

Puncturable Encryption

- Public key encryption with “update capabilities” on the secret key
- Secret key can be punctured on ciphertext s.t. this ciphertext can no longer be decrypted



Conventional encryption scheme:

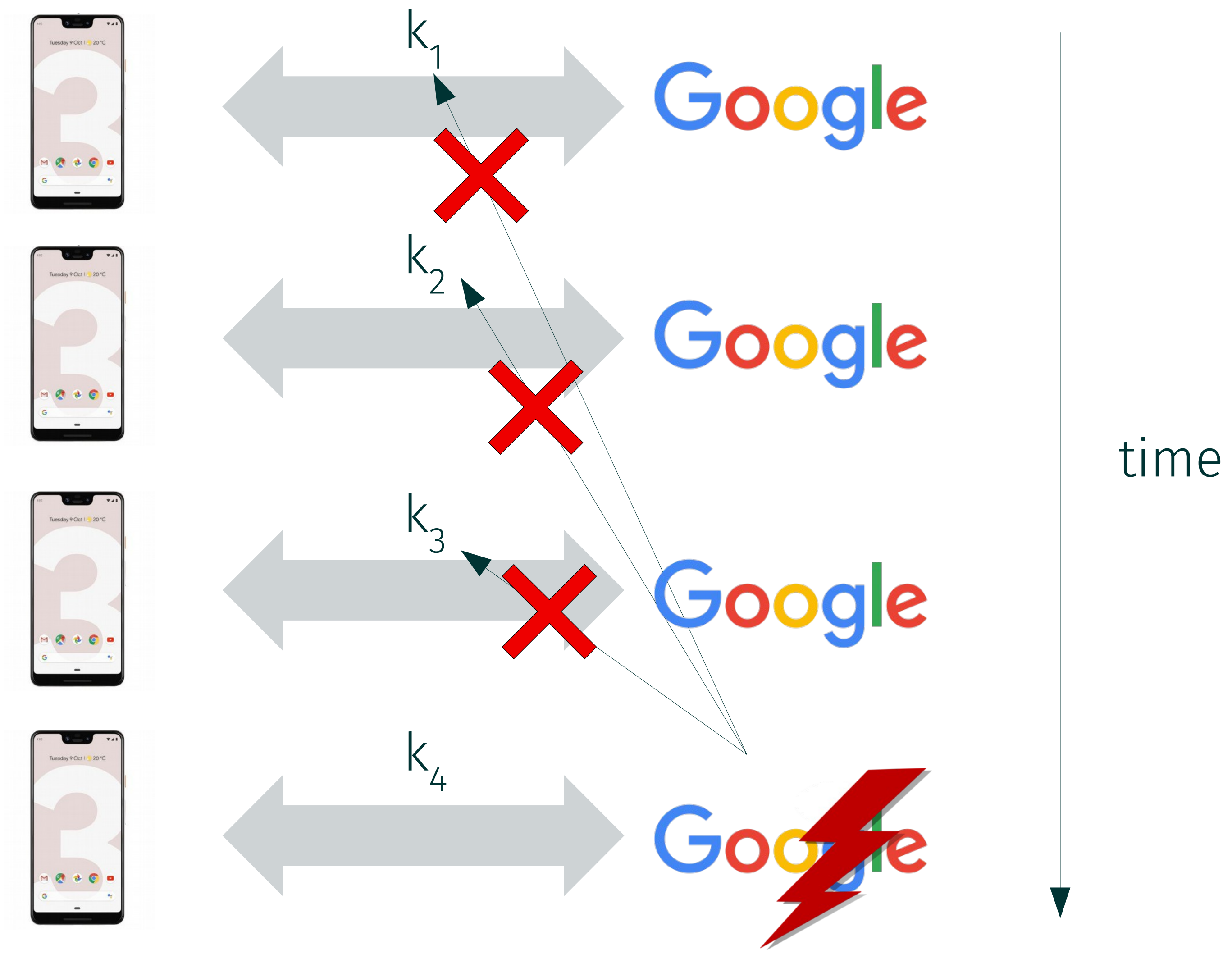
- (KeyGen, Enc, Dec)

+ Additional algorithm 🔑′ $\leftarrow$ Punc(🔑, C)

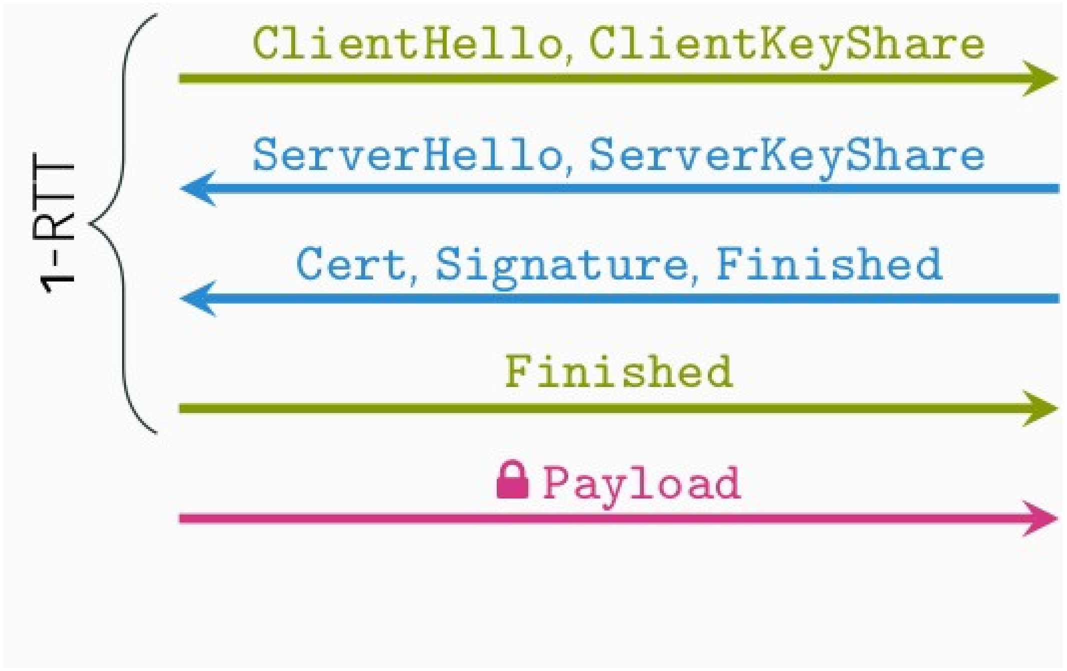
Properties

- 🔑′ no longer useful to decrypt C
- 🔑′ still useful to decrypt other ciphertexts
- Repeated puncturing possible

Perfect Forward Secrecy



Puncturable Encryption: Application fs 0-RTT Key Exchange



1-RTT
ClientHello, ClientKeyShare
ServerHello, ServerKeyShare
Cert, Signature, Finished
Finished
Payload

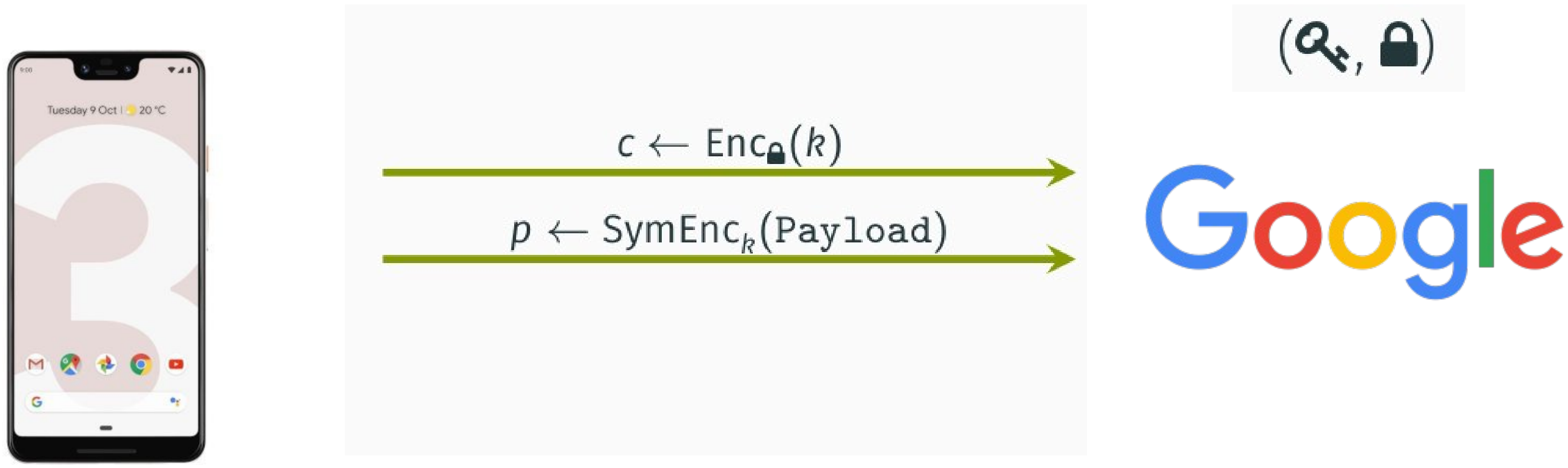
Google

Can we already send encrypted payload with the first message?

Desired properties:

- Replay protection
- Forward secrecy

Puncturable Encryption: Application fs 0-RTT Key Exchange

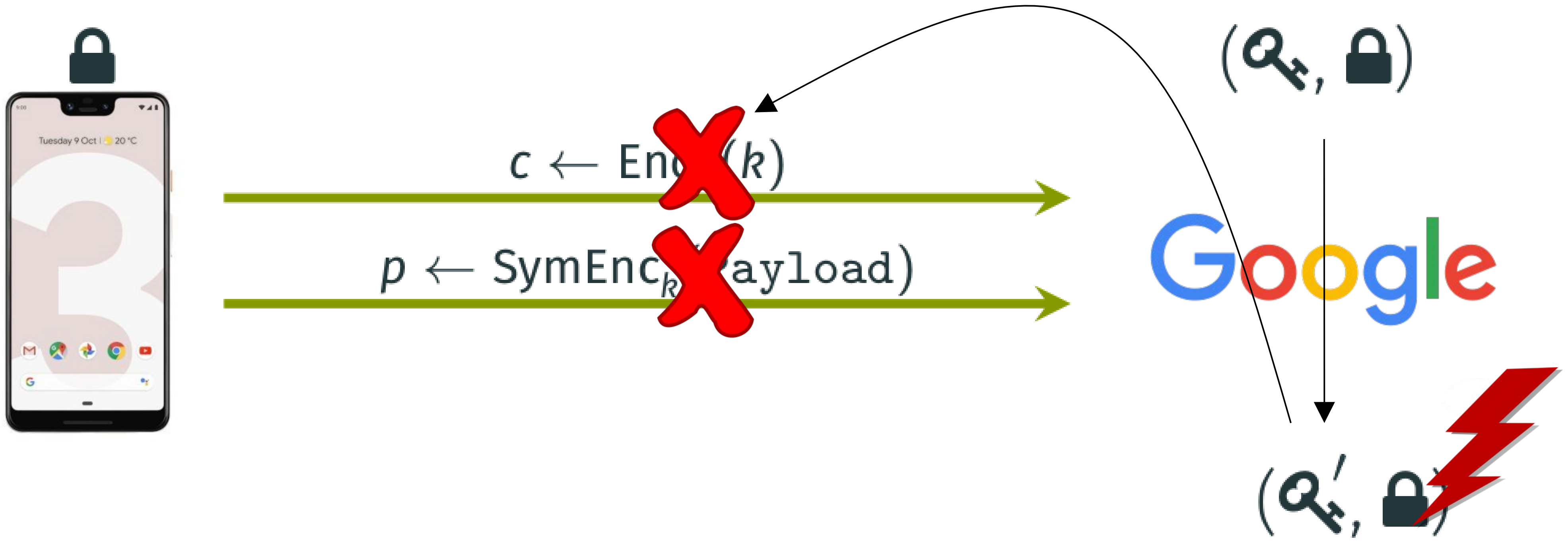

(🔑, 🔒)
$c \leftarrow \mathsf{Enc}_{🔒}(k)$
$p \leftarrow \mathsf{SymEnc}_k(\texttt{Payload})$
Google

Major deficiencies:

- No forward secrecy
- Vulnerable to replay attacks

Puncturable Encryption: Application fs 0-RTT Key Exchange

Use of **puncturable encryption** [GM15, GHJL17, DJSS18]



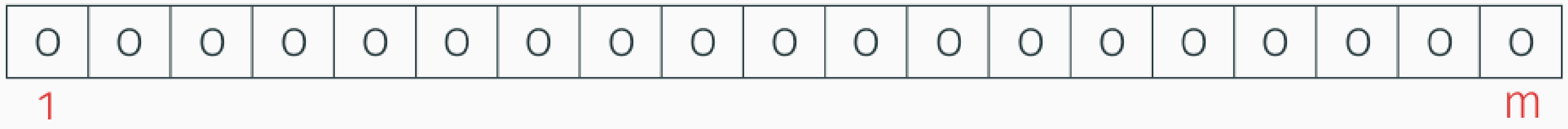
$c \leftarrow \mathsf{Enc}(k)$
$p \leftarrow \mathsf{SymEnc}_k(\mathtt{payload})$
Google

Puncturable Encryption

- We are looking at one construction idea
  - Construct a scheme with **non-negligible correctness error**: does not matter too much for key-exchange
    - E.g., 1 in 1000 sessions fail (can then fallback to 1-RTT)

- The most basic construction is called **Bloom Filter Encryption** (BFE)
  - Bloom Filter: data structure for probabilistic set membership checks

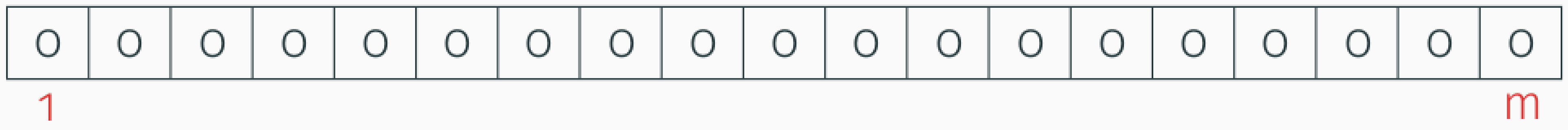
Puncturable Encryption: Bloom Filters



- Initial state $T := 0^m$
- k universal hash functions $(H_j)_{j \in [k]}$
- $H_j : \mathcal{U} \rightarrow [m]$
- Throughout this talk, let $k = 3$

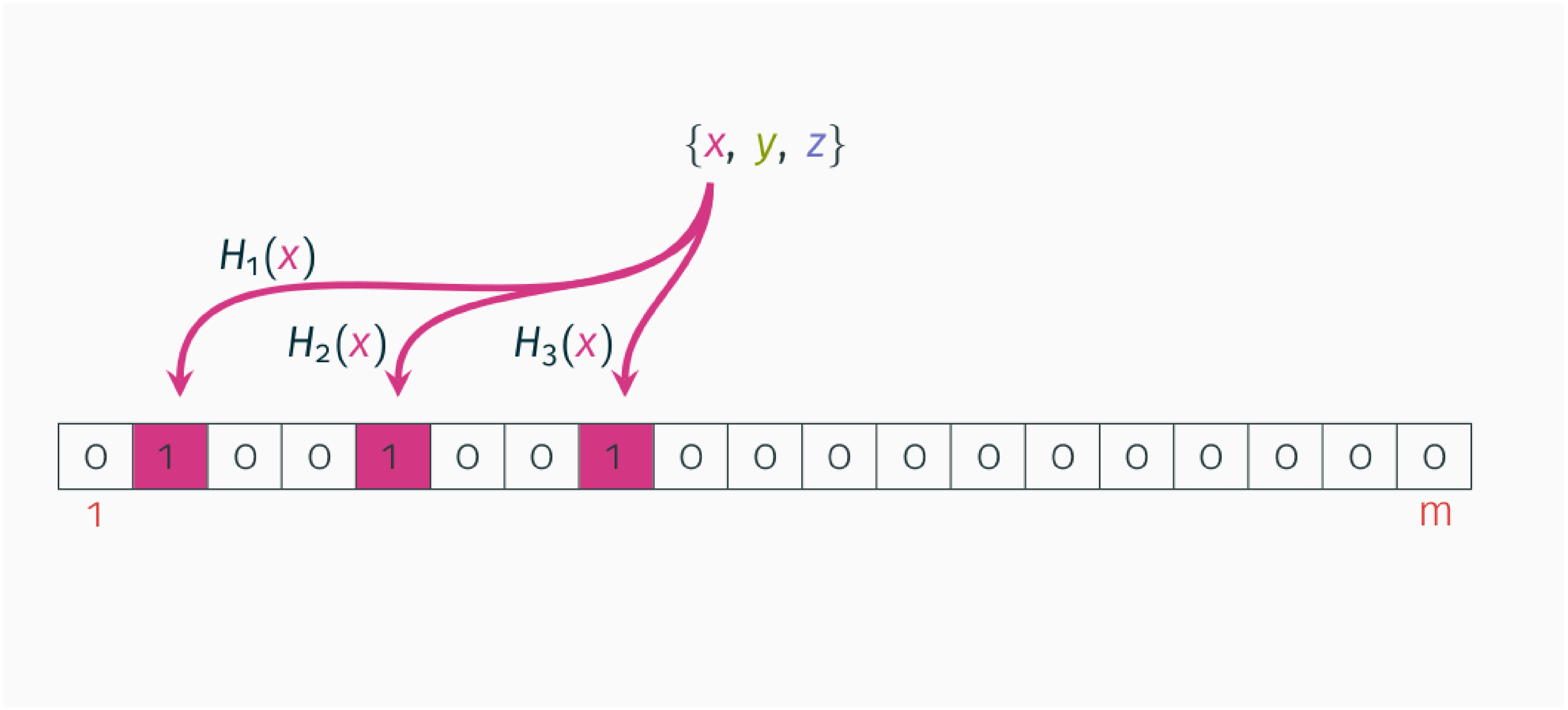
Puncturable Encryption: Bloom Filters

$\{x, y, z\}$

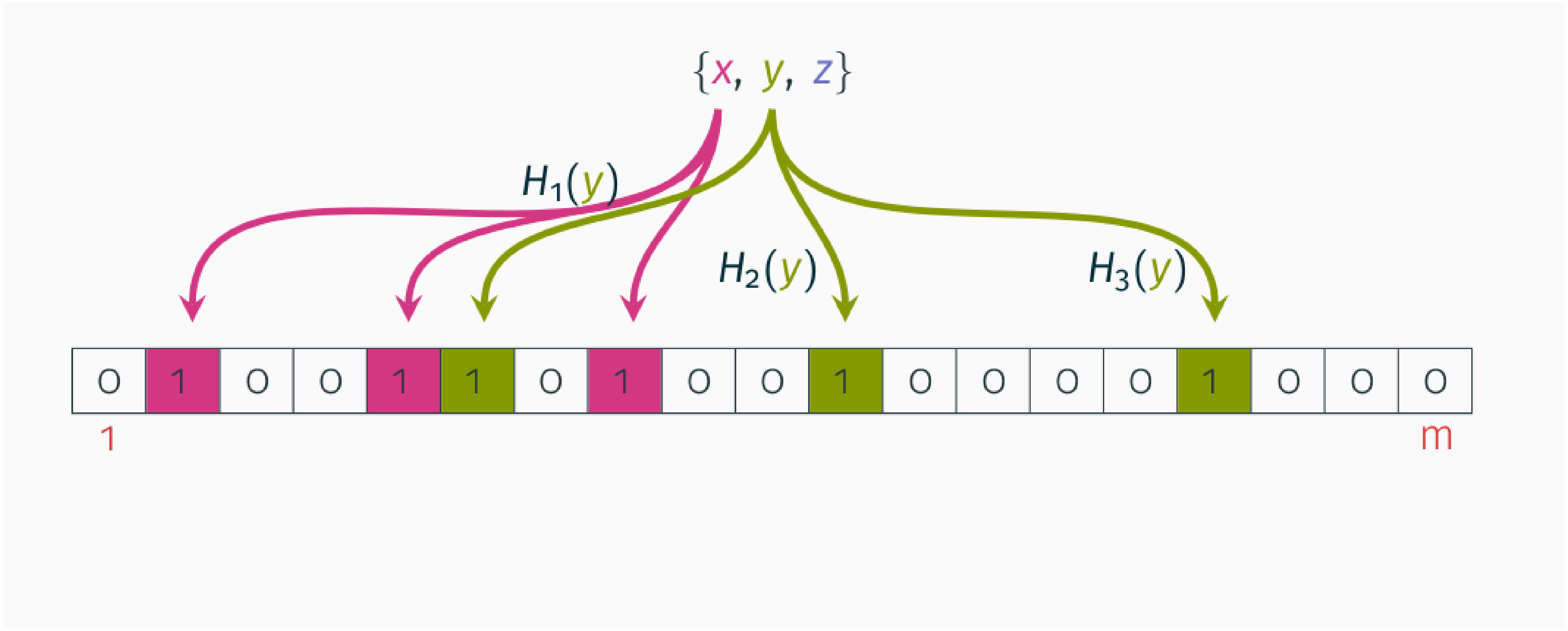


- Initial state $T := 0^m$
- k universal hash functions $(H_j)_{j\in[k]}$
- $H_j : \mathcal{U} \to [m]$
- Throughout this talk, let $k = 3$

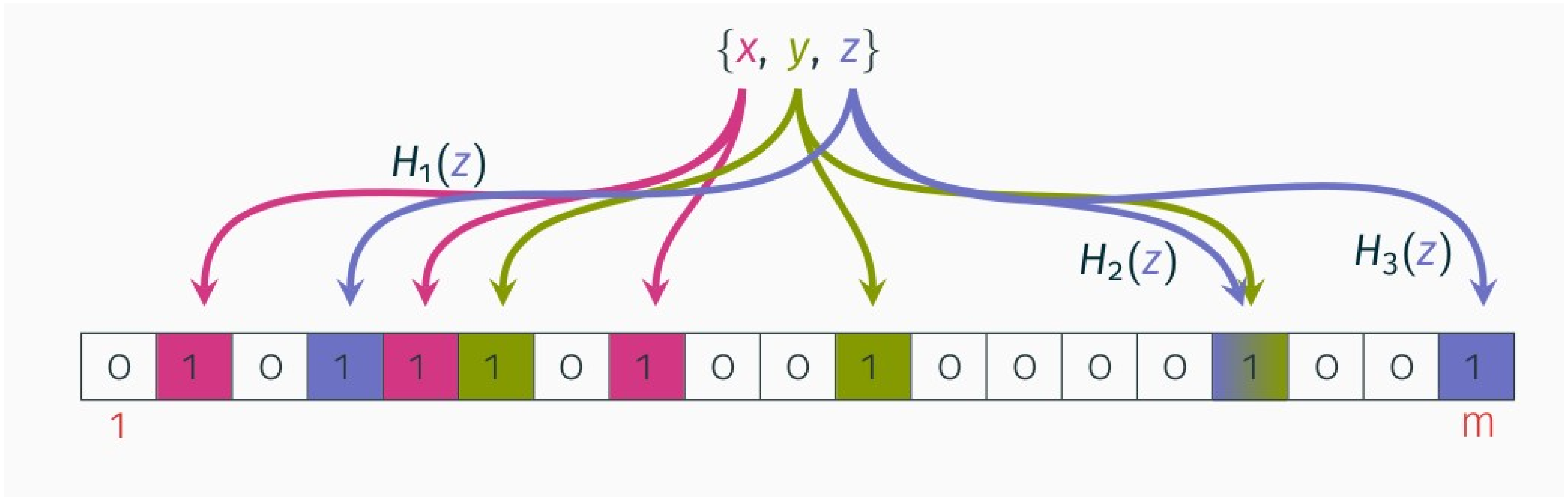
Puncturable Encryption: Bloom Filters



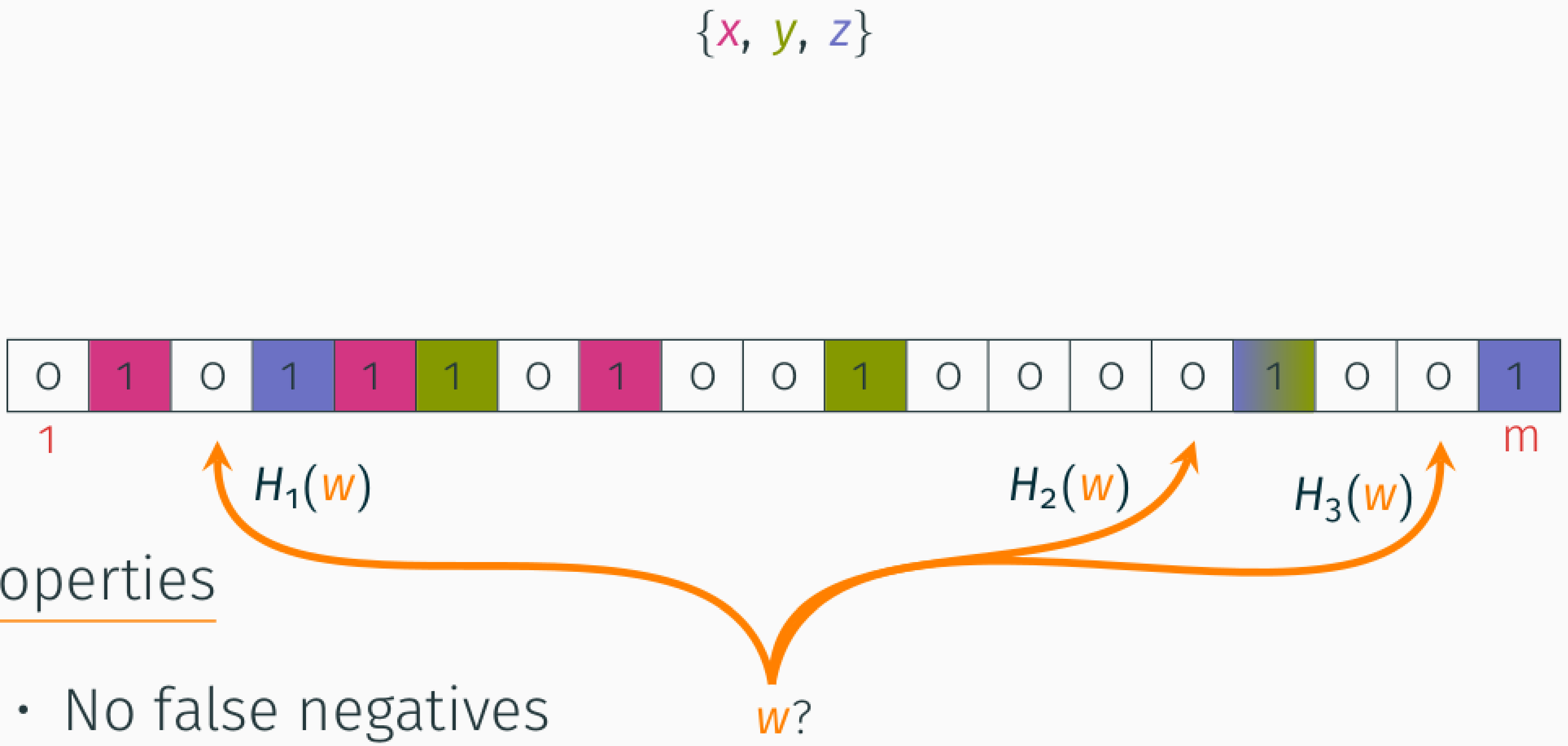
Puncturable Encryption: Bloom Filters



Puncturable Encryption: Bloom Filters



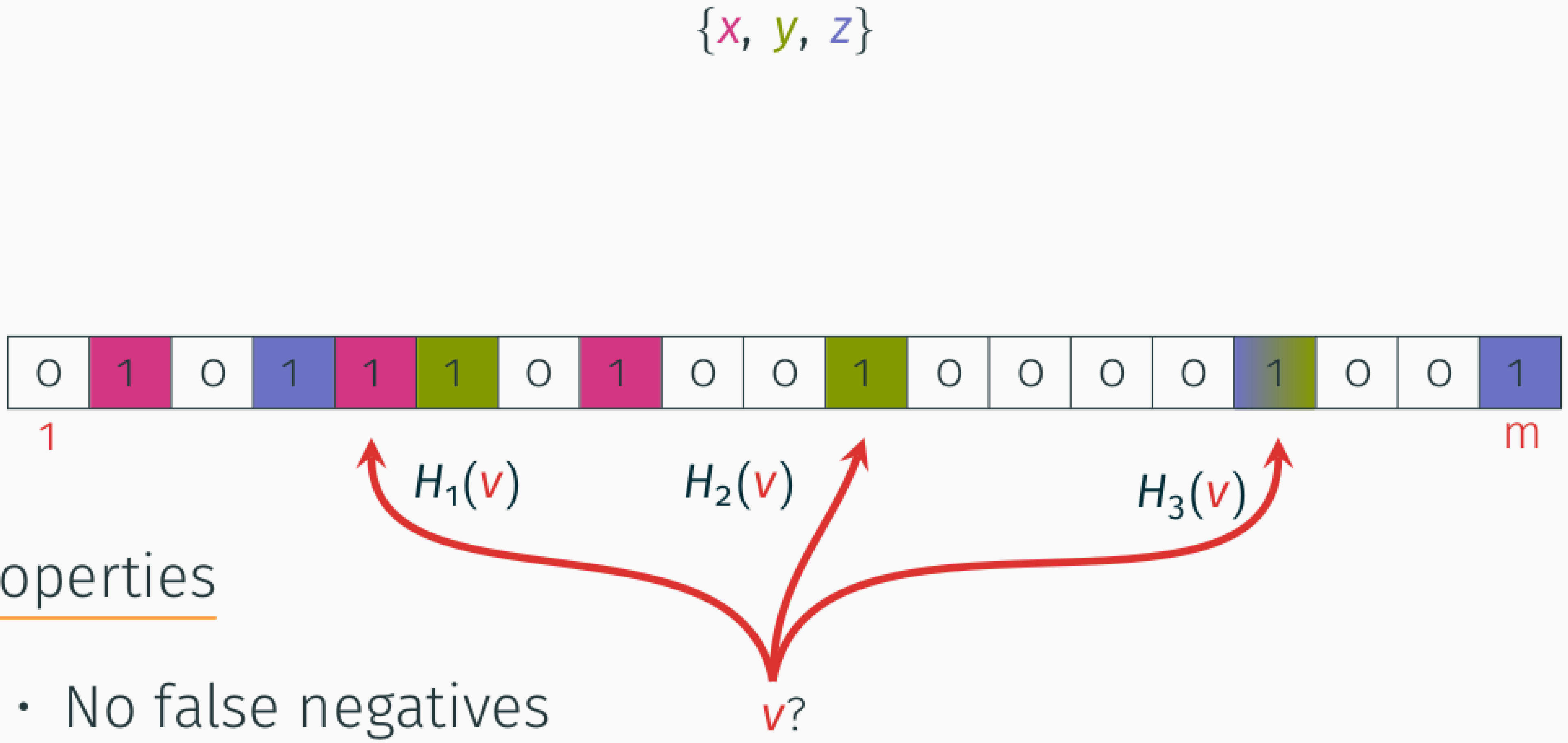
Puncturable Encryption: Bloom Filters



Properties

- No false negatives

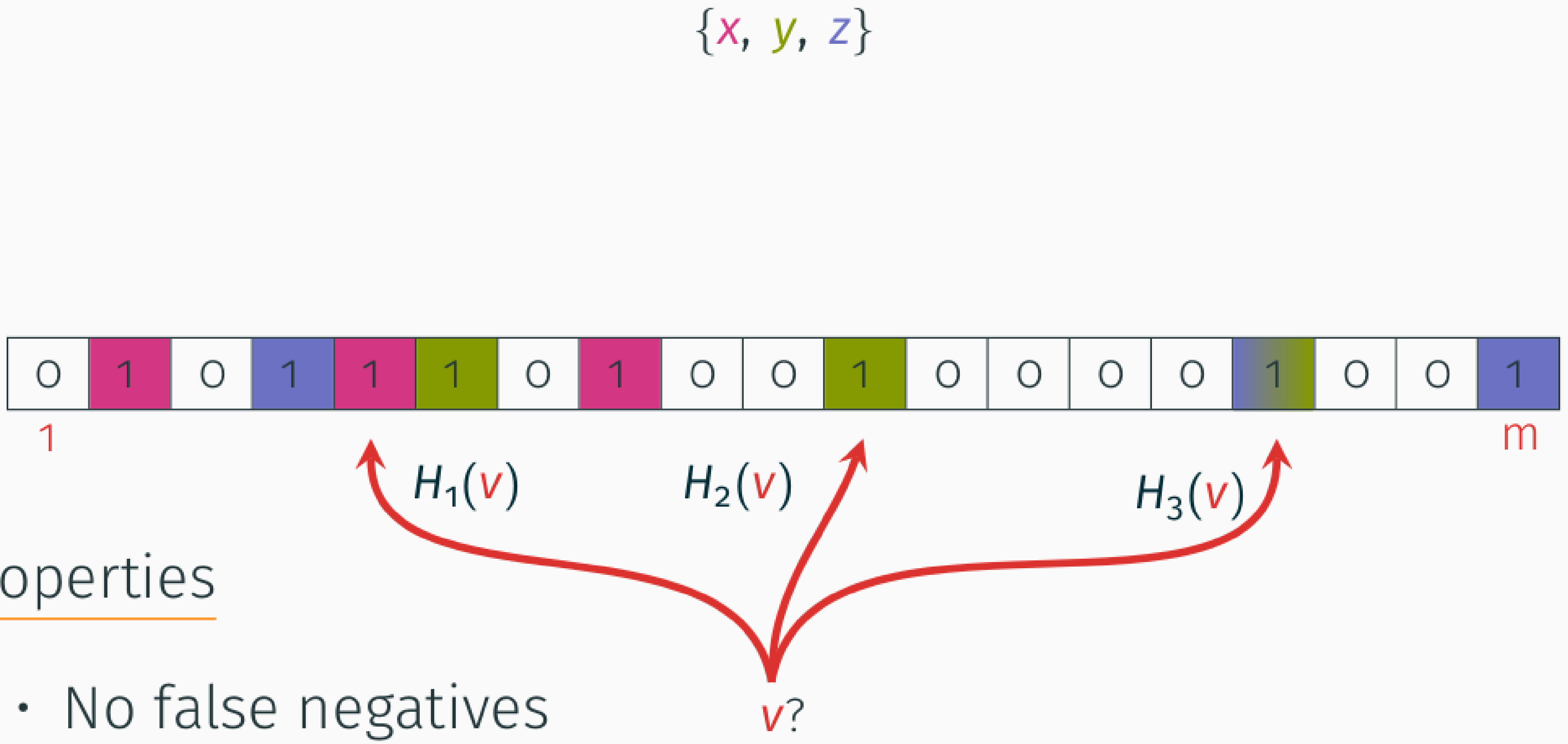
Puncturable Encryption: Bloom Filters



Properties

- No false negatives
- **False positives possible**

Puncturable Encryption: Bloom Filters



Properties

- No false negatives
- **False positives possible**
- Probability determined by k, m, and # inserted elements

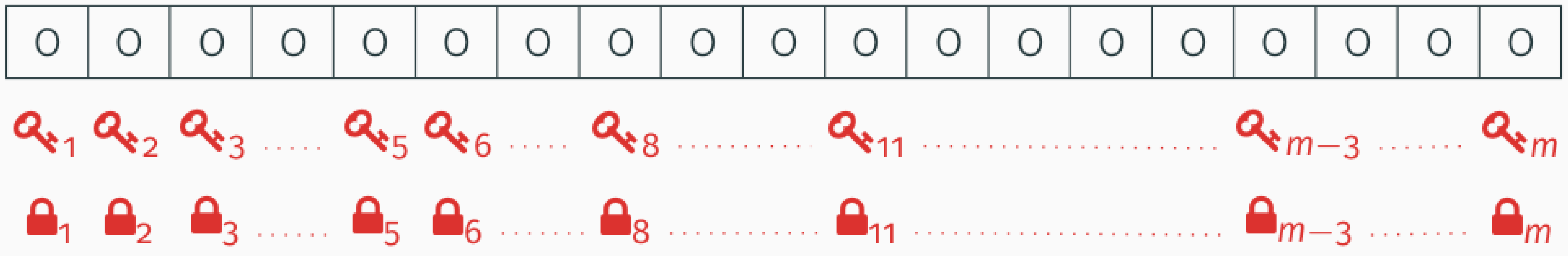
Puncturable Encryption: BFE Construction

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KeyGen

- Set up BF

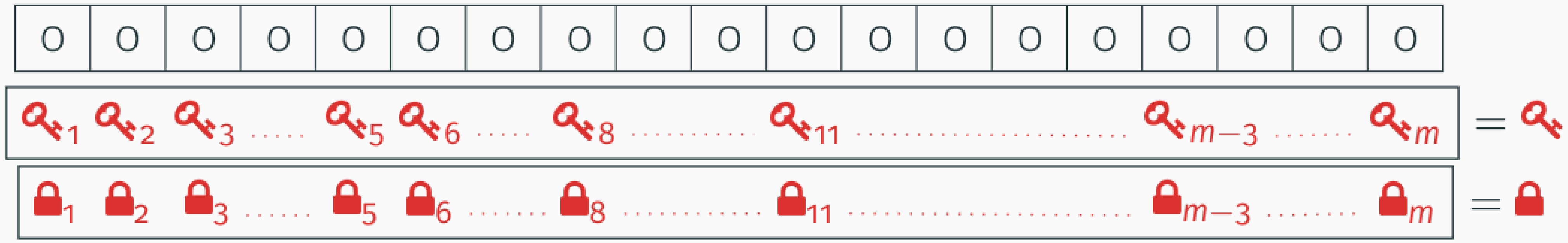
Puncturable Encryption: BFE Construction



KeyGen

- Set up BF
- Associate key pair to each bit

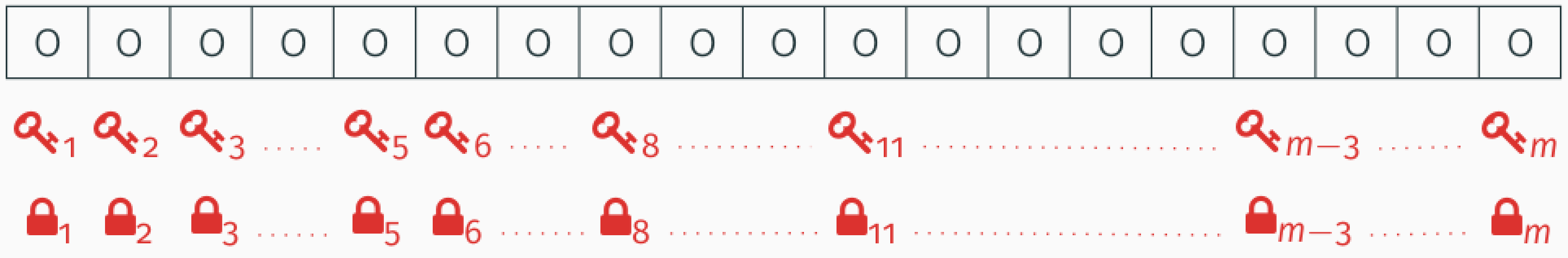
Puncturable Encryption: BFE Construction



KeyGen

- Set up BF
- Associate key pair to each bit
- Compose BFE key pair (🔑, 🔒)

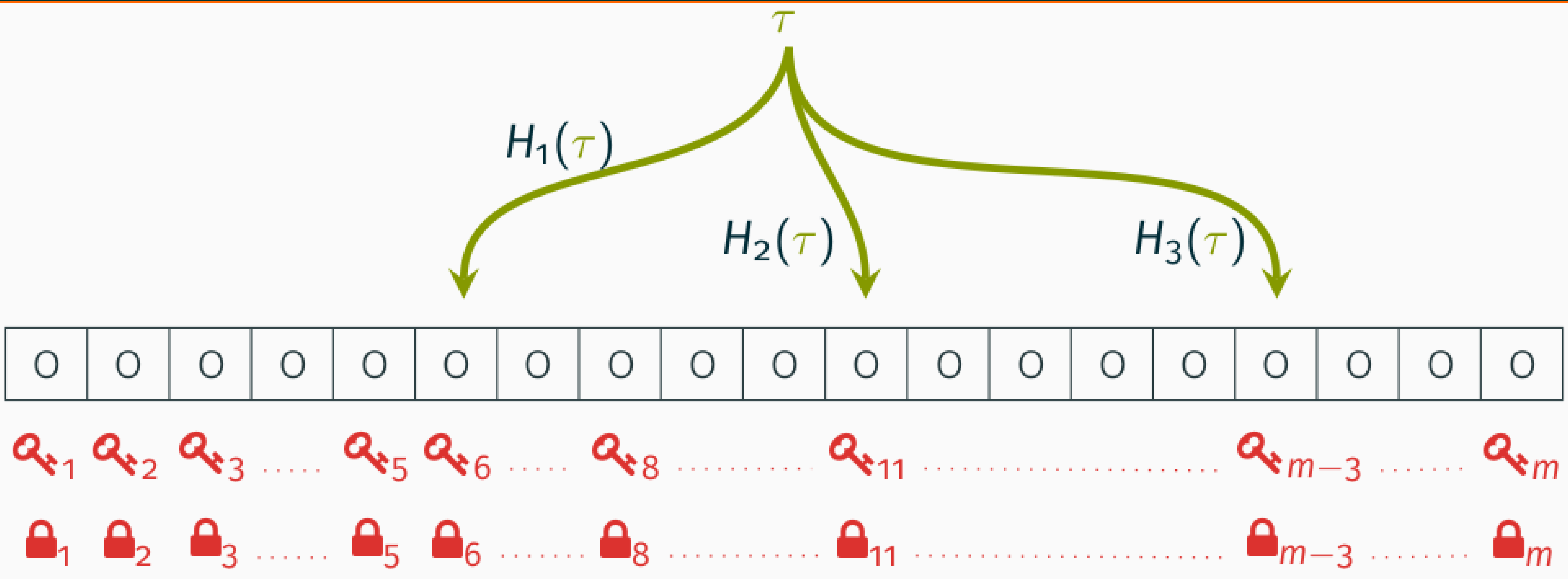
Puncturable Encryption: BFE Construction



Encrypt message M

- Randomly choose tag τ

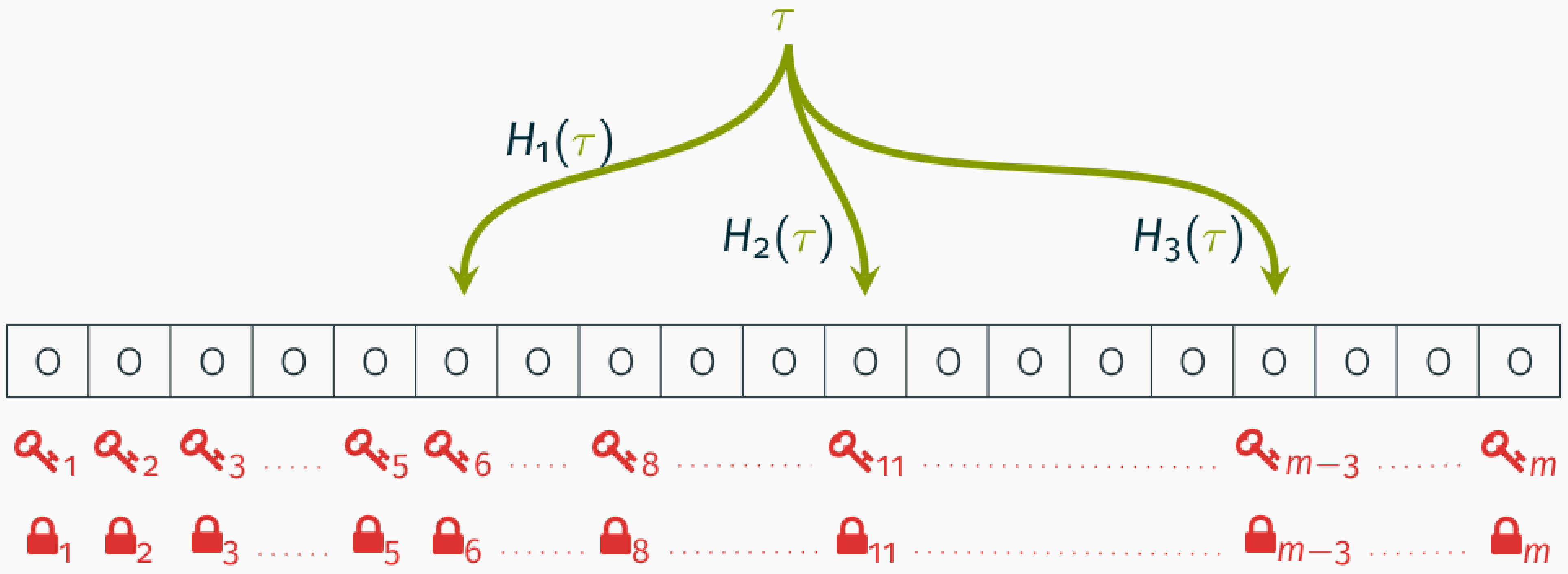
Puncturable Encryption: BFE Construction



Encrypt message M

- Randomly choose tag τ
- Determine indexes from τ

Puncturable Encryption: BFE Construction

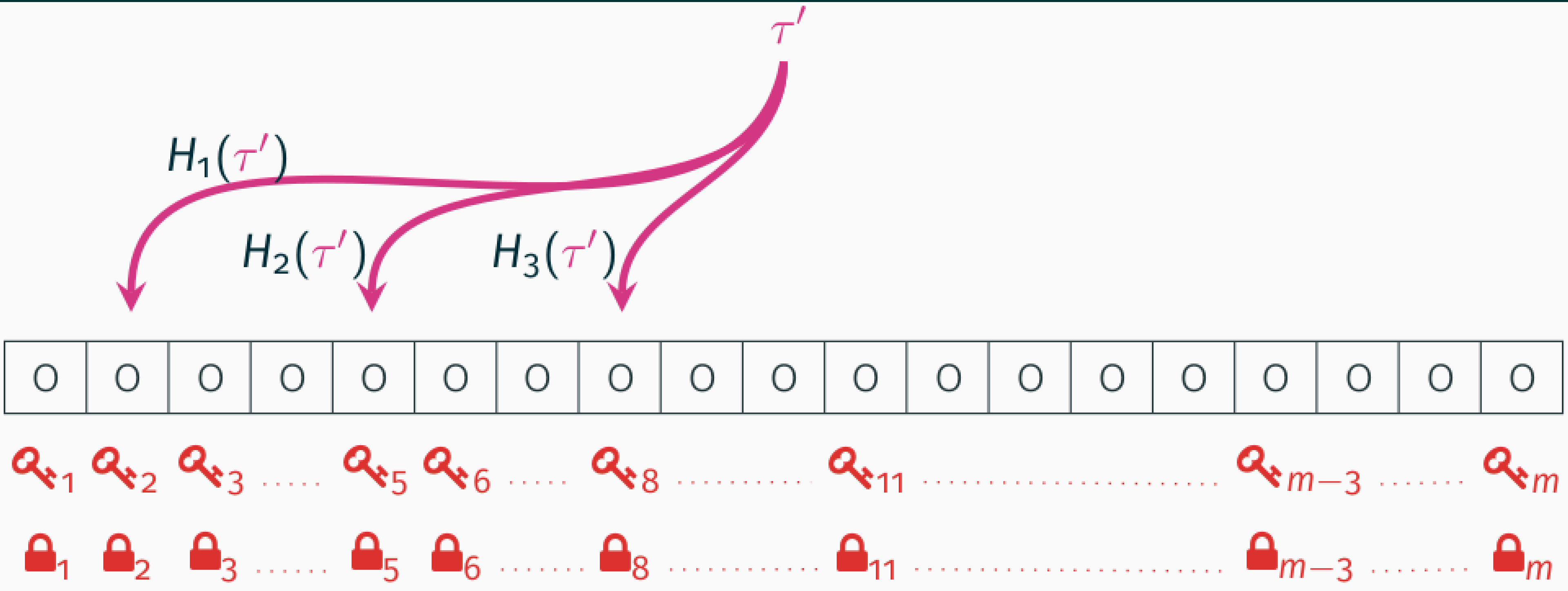


τ
$H_1(\tau)$
$H_2(\tau)$
$H_3(\tau)$
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 3 ..... 5 6 ..... 8 .... 11 .... $m-3$ .... m
1 2 3 ..... 5 6 ..... 8 .... 11 .... $m-3$ .... m

Encrypt message M

- Randomly choose tag τ
- Determine indexes from τ
- $C_\tau \leftarrow \mathsf{Enc}_{🔒_6 \vee 🔒_{11} \vee 🔒_{m-3}}(M)$

Puncturable Encryption: Bloom Filters



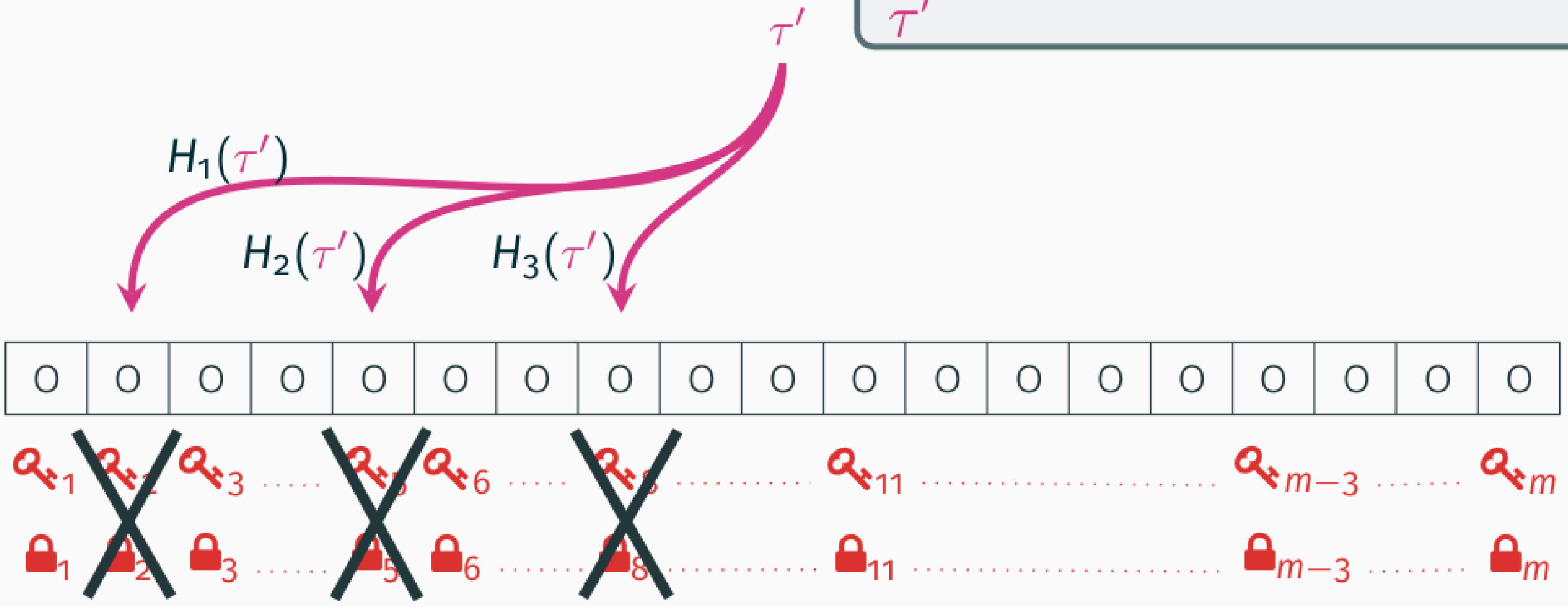
τ'
$H_1(\tau')$
$H_2(\tau')$
$H_3(\tau')$
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 3 …… 5 6 …… 8 ………… 11 ………………… m−3 …… m
1 2 3 …… 5 6 …… 8 ………… 11 ………………… m−3 …… m

Puncture ciphertext $C_{\tau'}$

- Determine BF indexes from τ'

Puncturable Encryption: Bloom Filters

Secret key no longer useful to decrypt $C_{\tau'}$ with associated tag τ'

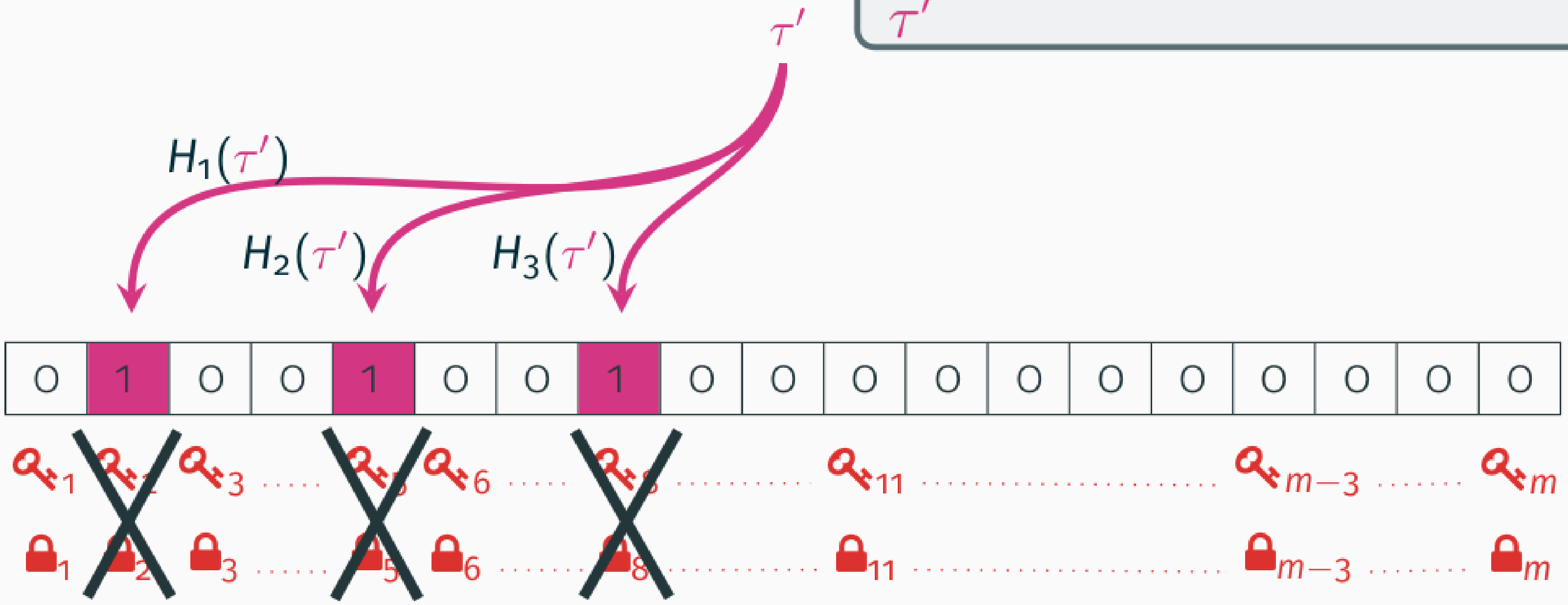


Puncture ciphertext $C_{\tau'}$

- Determine BF indexes from τ'
- Delete associated keys

Puncturable Encryption: Bloom Filters

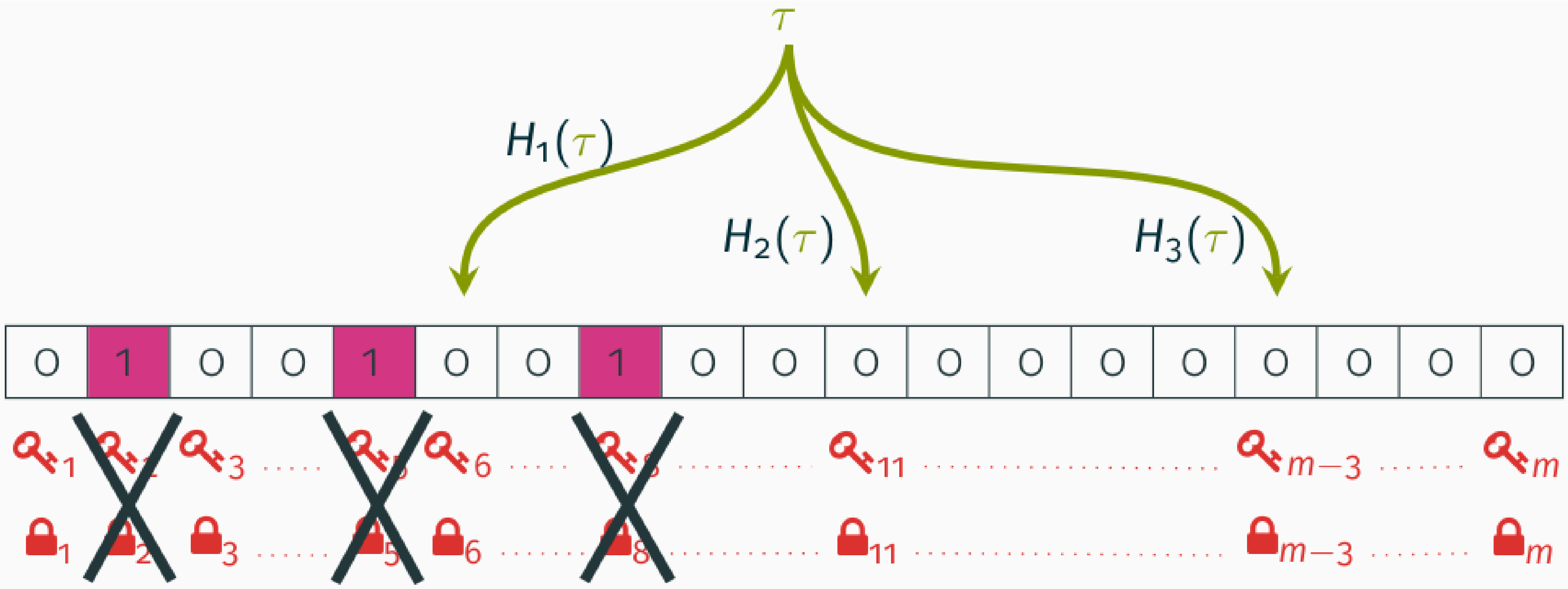
Secret key no longer useful to decrypt $C_{\tau'}$ with associated tag τ'



Puncture ciphertext $C_{\tau'}$

- Determine BF indexes from τ'
- Delete associated keys
- Update BF state

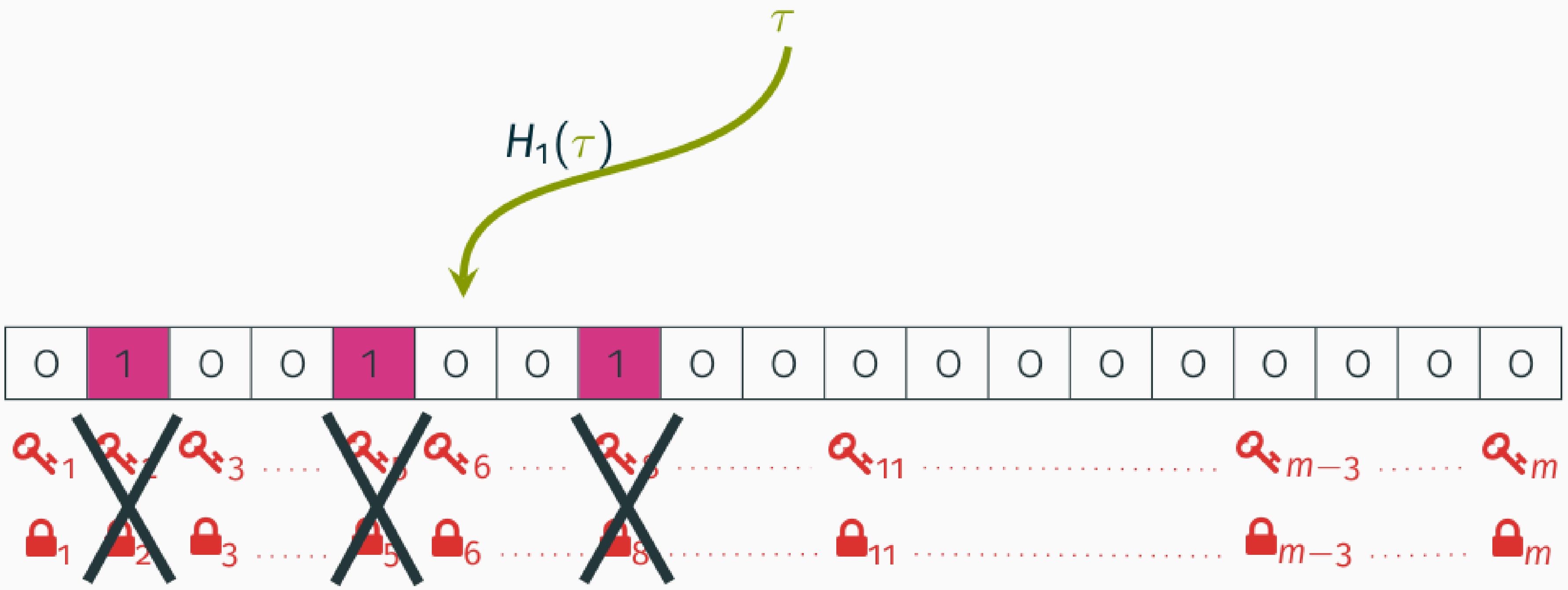
Puncturable Encryption: Bloom Filters



Decrypt ciphertext C_τ

- Determine BF indexes from τ

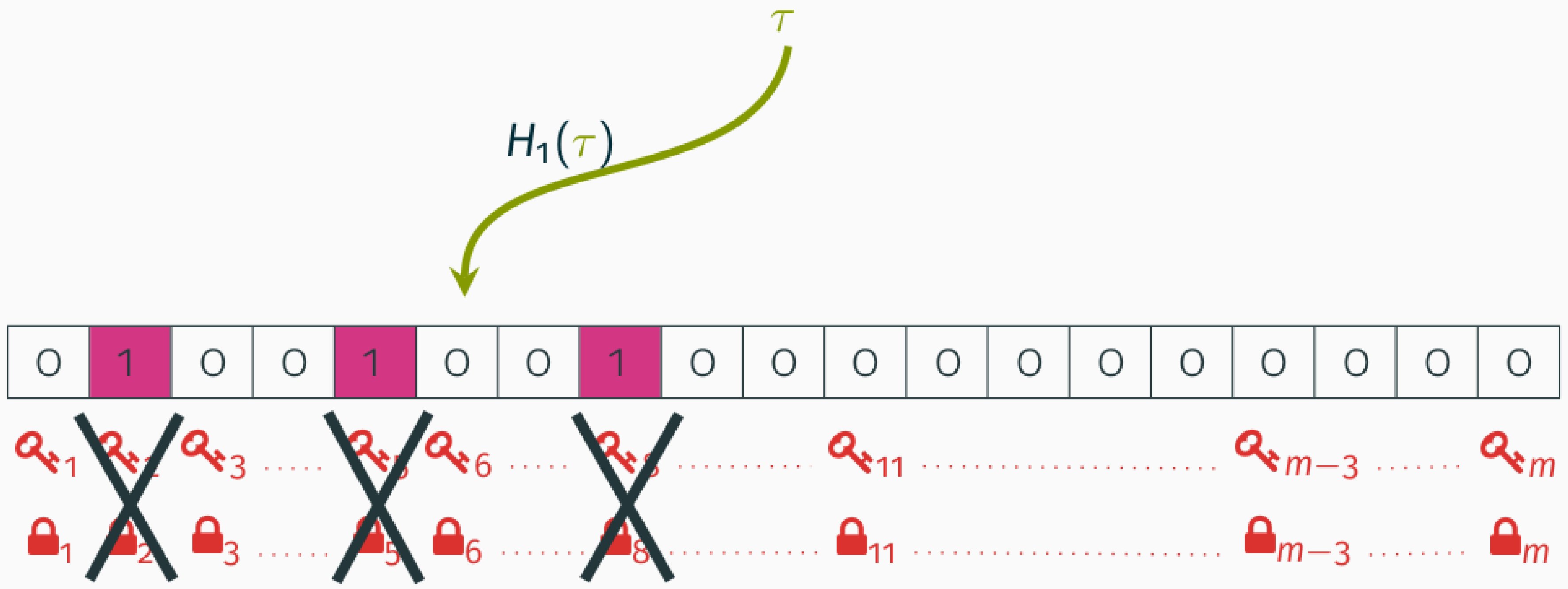
Puncturable Encryption: Bloom Filters



Decrypt ciphertext C_τ

- Determine BF indexes from τ
- Let i lowest index w. $BF[i] = 0$

Puncturable Encryption: Bloom Filters



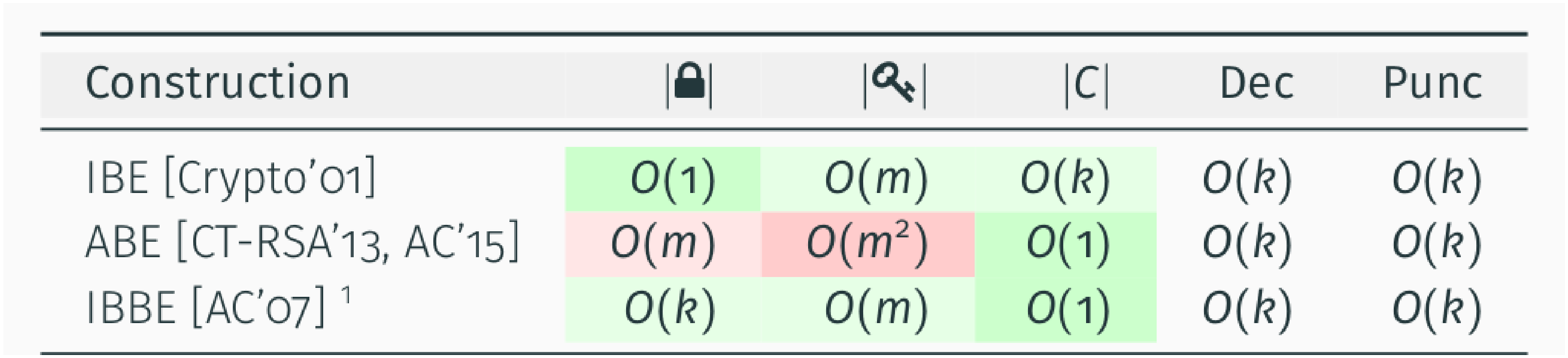
Decrypt ciphertext C_τ

- Determine BF indexes from τ
- Let i lowest index w. $BF[i] = 0$
- $M \leftarrow \mathsf{Dec}_{6}(C_\tau)$

Puncturable Encryption: Bloom Filters

- Maximum # of elements in BF: 2^{20}
  - ≈ 2^{12} puncturings/day for full year
- False positive probability: 10^{-3}
- BF size $m = n \cdot \ln p/(\ln 2)^2 \approx$ 2MB
- # hash functions $k = \lceil m/n \cdot \ln 2 \rceil = 10$

- Constructions from different primitives
  - Identity-based encryption (IBE), Attribute-based encryption (ABE)
  - Identity-based broadcast encryption (IBBE)



Construction	|🔒|	|🔑|	|C|	Dec	Punc
IBE [Crypto'01]	$O(1)$	$O(m)$	$O(k)$	$O(k)$	$O(k)$
ABE [CT-RSA'13, AC'15]	$O(m)$	$O(m^2)$	$O(1)$	$O(k)$	$O(k)$
IBBE [AC'07] [1]	$O(k)$	$O(m)$	$O(1)$	$O(k)$	$O(k)$

The End

- Thank you all for participating in the course! It was a lot of fun!
- If you are interested in summer internships/bachelor/master projects please just contact me

Good luck for the final exam!!