From Hard to Moderately-hard

New Frontiers For Cryptography

Esteban Landerreche
CWI Scientific Meeting
17 April 2020

Encryption

Sender and receiver can privately communicate

Digital Signatures

Bind a public key to a message

Hashing

Create a short pseudo-random message fingerprint

Breaking cryptographic designs should be *hard* functions

Number-theoretic Problems Factorization, discrete logarithms

NP-hard Problems Shortest lattice vector, decoding random linear codes

Symmetric Cryptography Finding the secret key

Time-lock cryptography Iterated squarings

Proofs of Work Brute force search

Challenger

- Set $n \leftarrow pq$ with p, q randomly chosen primes
- Encrypt message with key K
- Choose random $a \in \mathbb{Z}_n$ and compute $c \leftarrow a^{2^T} \bmod n$ for sufficiently large T
- Release encrypted message, $K \oplus c$, T, n and a

Challenger

- Set $n \leftarrow pq$ with p, q randomly chosen primes
- Encrypt message with key K
- Choose random $a \in \mathbb{Z}_n$ and compute $c \leftarrow a^{2^T} \bmod n$ for sufficiently large T
- Release encrypted message, $K \oplus c$, T, n and a

Solver

- Solver computes $a^{2^T} \bmod n$ and adds it to $K \oplus c$
- Uses K to decrypt the message

Challenger

- Set $n \leftarrow pq$ with p, q randomly chosen primes
- Encrypt message with key K
- Choose random $a \in \mathbb{Z}_n$ and compute $c \leftarrow a^{2^T} \bmod n$ for sufficiently large T
- Release encrypted message, $K \oplus c$, T, n and a

Solver

- Solver computes $a^{2^T} \bmod n$ and adds it to $K \oplus c$
- Uses K to decrypt the message

Challenger

- Computes $\phi(n) \leftarrow (p-1)(q-1)$
- Sets $e \leftarrow 2^T \bmod \phi(n)$
- Computes $a^e \bmod n$

Challenger

- Computes $\phi(n) \leftarrow (p-1)(q-1)$
- Sets $e \leftarrow 2^T \bmod \phi(n)$
- Computes $a^e \bmod n$

Solver

- Computes $a_1 \leftarrow a^2 \bmod n$

Challenger

- Computes $\phi(n) \leftarrow (p-1)(q-1)$
- Sets $e \leftarrow 2^T \bmod \phi(n)$
- Computes $a^e \bmod n$

Solver

- Computes $a_1 \leftarrow a^2 \bmod n$
- Computes $a_2 \leftarrow a_1^2 \bmod n$

Challenger

- Computes $\phi(n) \leftarrow (p-1)(q-1)$
- Sets $e \leftarrow 2^T \bmod \phi(n)$
- Computes $a^e \bmod n$

Solver

- Computes $a_1 \leftarrow a^2 \bmod n$
- Computes $a_2 \leftarrow a_1^2 \bmod n$

$\ldots$

Challenger

- Computes $\phi(n) \leftarrow (p-1)(q-1)$
- Sets $e \leftarrow 2^T \bmod \phi(n)$
- Computes $a^e \bmod n$

Solver

- Computes $a_1 \leftarrow a^2 \bmod n$
- Computes $a_2 \leftarrow a_1^2 \bmod n$

  $\dots$
- Computes $a_T \leftarrow a_{T-1}^2 \bmod n$

- Given
  - Target T
  - Hash function H
  - Message m

find a bitstring r such that $H(m||r) < T$

- Given
  - Target T
  - Hash function H
  - Message m

  find a bitstring r such that $H(m||r) < T$
- If H is cryptographically secure, the only way to do this is through brute force
- Expected amount of work is $T/2^n$ where n is the output hash bit size

- Given
  - Target T
  - Hash function H
  - Message m

  find a bitstring r such that $H(m||r) < T$
- If H is cryptographically secure, the only way to do this is through brute force
- Expected amount of work is $T/2^n$ where n is the output hash bit size
- In contrast to time-lock, very parallelizable

Enter Bitcoin

- Distributed ledger maintained by an unpermissioned network of parties
- Uses proofs of work to provide a notion of *identity*
- Achieves state machine replication
- Not impossible to disrupt, just hard and with a high cost

- Bitcoin is a chain of blocks of transactions
- Users must create a block that is a valid proof of work to add it to the chain
- In order to *rewrite* the chain, one must find a new proof of work for each block
- Parameters are tuned such that a block will be created every 10 minutes

- Bitcoin is a chain of blocks of transactions
- Users must create a block that is a valid proof of work to add it to the chain
- In order to *rewrite* the chain, one must find a new proof of work for each block
- Parameters are tuned such that a block will be created every 10 minutes
- Bitcoin is a timestamp server

New Setting

New assumptions Minimal setup

New goals Public verifiability, security under incentive compatibility

New primitives Moderately-hard functions, proof-of-resource, NIZK

A Protocol in this New Setting

- First achieved by [HS91]
- Most protocols are based on hashchains
- Requires online verification

Backdating Security (informal)

A timestamping scheme is backdating secure if an adversary cannot claim something was created earlier than it was.

Postdating Security (informal)

A timestamping scheme is postdating secure if an adversary cannot claim something was created later than it was.

Non-interactive Timestamping

- Impossibility result for non-interactive timestamping
- Simulation of an honest prover

- Impossibility result for non-interactive timestamping
- Simulation of an honest prover

We are in a new setting

- Impossibility result for non-interactive timestamping
- Simulation of an honest prover

We are in a new setting

- Achievable with a moderately-hard function (verifiable delay function) [LSS20]

Inverted time-lock puzzles

Inverted time-lock puzzles

Prover

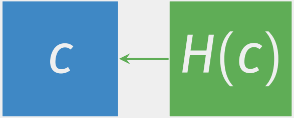
Computes a function which takes T sequential steps and outputs the result next to a proof π

Verifier

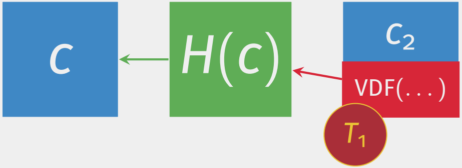
Can efficiently check whether the computation was done correctly using the proof π

Our Construction

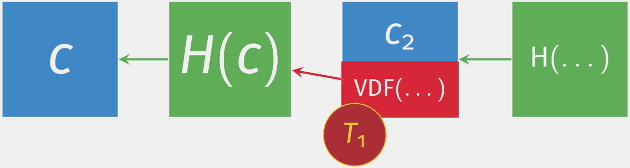
c


c
H(c)

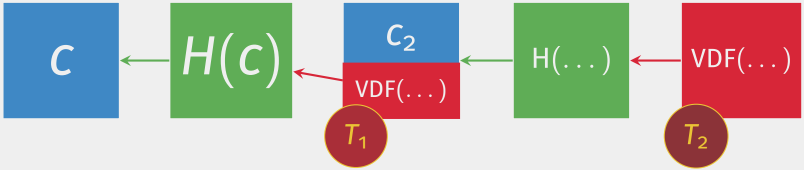
Our Construction

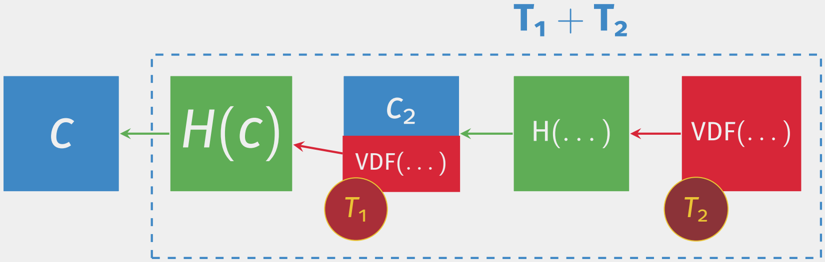


Our Construction



Our Construction





$T_1 + T_2$
c
$H(c)$
c_2
VDF(...)
H(...)
VDF(...)
T_1
T_2

Theorem (Security of the Protocol [LSS20])

If an adversary has corrupted the prover T time ago and has an advantage of $\alpha \geq 1$ in VDF computation then:

- it cannot modify any record marked older than $T \cdot \alpha$
- it can either keep all records marked older than $T \cdot \alpha$ or none
- any modified record of created A time ago has timestamp $< A \cdot \alpha$ ago.

New Directions

- New setting which allows us to do what we couldn't before
- Exisiting frameworks need to be extended to accommodate for them
- We created backdating-secure protocol in the UC framework where an adversary has a time dilution factor α

Thank you

References

Stuart Haber and W. Scott Stornetta.
How to time-stamp a digital document.
Journal of Cryptology, 3(2):99–111, Jan 1991.

Markus Jakobsson and Ari Juels.
Proofs of work and bread pudding protocols.
In *Secure information networks*, pages 258–272. Springer, 1999.

Esteban Landerreche, Marc Stevens, and Christian Schaffner.
Non-interactive cryptographic timestamping based on verifiable delay functions.
In *International Conference on Financial Cryptography and Data Security*. Springer, 2020.

Ronald L Rivest, Adi Shamir, and David A Wagner.
Time-lock puzzles and timed-release crypto.
1996.