FROM HARD TO MODERATELY-HARD

NEW FRONTIERS FOR CRYPTOGRAPHY

ESTEBAN LANDERRECHE
CWI SCIENTIFIC MEETING
17 APRIL 2020
Cryptography in Practice

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 $c \leftarrow a^{2T} \mod n$ for sufficiently large $T$
- Release encrypted message, $K \oplus c$, $T$, $n$ and $a$
## Time-lock Puzzles

### 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 $c \leftarrow a^{2^T} \mod n$ for sufficiently large $T$
- Release encrypted message, $K \oplus c$, $T$, $n$ and $a$

### Solver

- Solver computes $a^{2^T} \mod n$ and adds it to $K \oplus c$
- Uses $K$ to decrypt the message
**Time-lock Puzzles**

<table>
<thead>
<tr>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set $n \leftarrow pq$ with $p, q$ randomly chosen primes</td>
</tr>
<tr>
<td>Encrypt message with key $K$</td>
</tr>
<tr>
<td>Choose random $a \in \mathbb{Z}_n$ and compute $c \leftarrow a^{2^T} \mod n$ for sufficiently large $T$</td>
</tr>
<tr>
<td>Release encrypted message, $K \oplus c$, $T$, $n$ and $a$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver computes $a^{2^T} \mod n$ and adds it to $K \oplus c$</td>
</tr>
<tr>
<td>Uses $K$ to decrypt the message</td>
</tr>
</tbody>
</table>
Challenger

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

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

Solver

- Computes $a_1 \leftarrow a^2 \mod n$
**Time-lock Puzzles**

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

### Solver
- Computes $a_1 \leftarrow a^2 \mod n$
- Computes $a_2 \leftarrow a_1^2 \mod n$
### Time-lock Puzzles

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

**Solver**
- Computes $a_1 \leftarrow a^2 \mod n$
- Computes $a_2 \leftarrow a_1^2 \mod n$
  
  ...
## Time-lock Puzzles

### Challenger
- Computes \( \phi(n) \leftarrow (p - 1)(q - 1) \)
- Sets \( e \leftarrow 2^T \mod \phi(n) \)
- Computes \( a^e \mod n \)

### Solver
- Computes \( a_1 \leftarrow a^2 \mod n \)
- Computes \( a_2 \leftarrow a_1^2 \mod n \)
  
  \[ \ldots \]
- Computes \( a_T \leftarrow a_{T-1}^2 \mod n \)
Given

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

Find a bitstring $r$ such that $H(m||r) < T$
PROOF OF WORK

- 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 $\frac{T}{2^n}$ where $n$ is the output hash bit size.

In contrast to time-lock, very parallelizable.
- 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.
- Impossibility result for non-interactive timestamping
- Simulation of an honest prover
Non-Interactive Timestamping

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

We are in a new setting
Non-interactive Timestamping

- 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 $\pi$

**Verifier**
Can efficiently check whether the computation was done correctly using the proof $\pi$
Our Construction

\[ C \rightarrow H(C) \]
Our Construction

\[
C \xrightarrow{H(C)} C_2 \xleftarrow{\text{VDF}(\ldots)} T_1
\]
Our Construction

C \rightarrow H(C) \rightarrow C_2 \rightarrow VDF(\ldots) \rightarrow H(\ldots)
Our Construction

\[ C \quad H(C) \quad C_2 \quad H(...) \quad \text{VDF}(...) \]

\[ T_1 \quad T_2 \]
\[ T_1 + T_2 \]

Diagram:

- C
- \( H(C) \)
- \( C_2 \)
- \( VDF(\ldots) \)
- \( T_1 \)
- \( T_2 \)
- \( T_{\text{one.osf}} + T_{\text{two.osf}} \)
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
- Existing 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 $\alpha$
Stuart Haber and W. Scott Stornetta.  
**How to time-stamp a digital document.**  

Markus Jakobsson and Ari Juels.  
**Proofs of work and bread pudding protocols.**  

Esteban Landerreche, Marc Stevens, and Christian Schaffner.  
**Non-interactive cryptographic timestamping based on verifiable delay functions.**  

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