T/Key: Second-Factor Authentication Without Server Secrets

Dima Kogan[1], Nathan Manohar[2], Dan Boneh[1]

[1]Stanford, [2]UCLA

Passwords have multiple security issues

eavesdropping/key logging

phishing

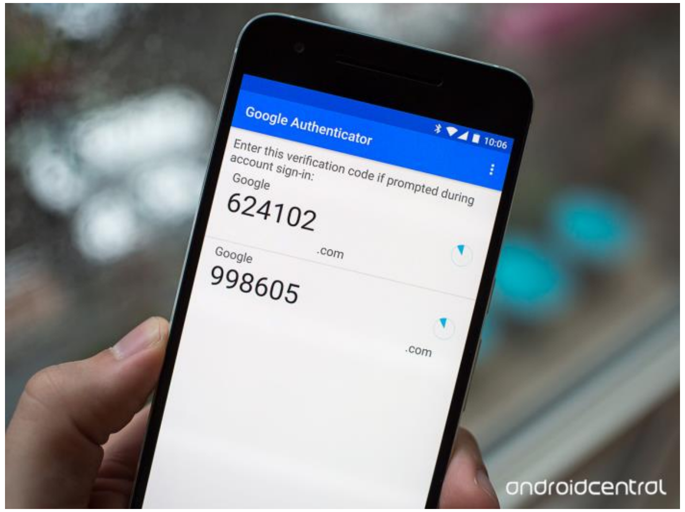
password reuse

Two-factor authentication

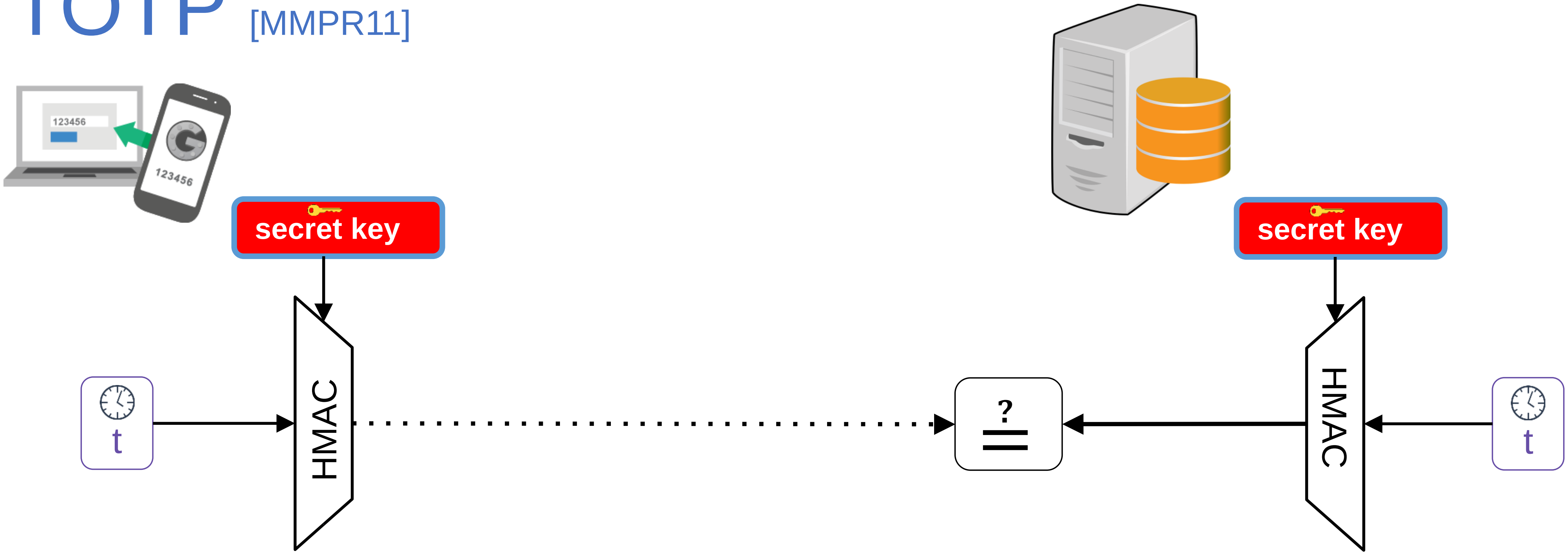
- Something you know
- Something you have

TOTP (time-based one-time password)

- User registers by scanning a QR code
- User logs in by copying an OTP

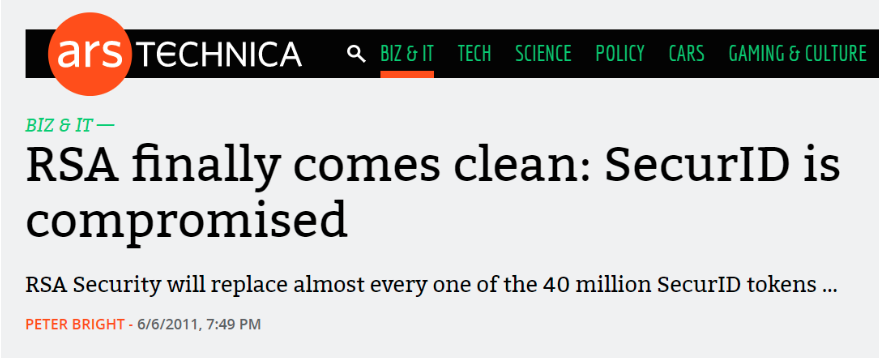


TOTP [MMPR11]



The problem with TOTP: secrets on the server

- Verifying the TOTP code requires the same secret as to generate it
- A **one time attack** on the server gives the attacker **persistent access**



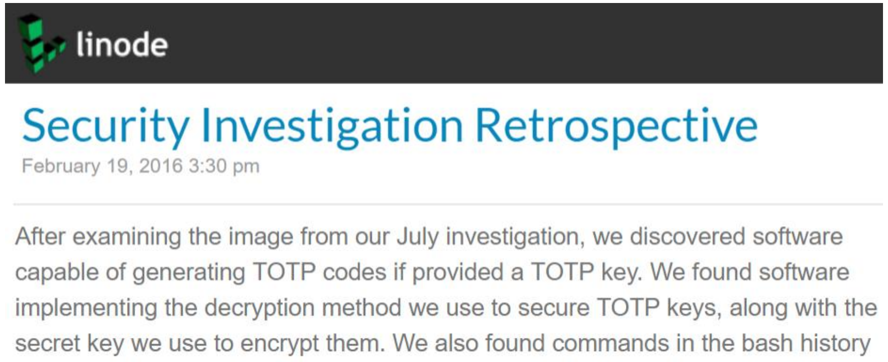
ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

BIZ & IT —

RSA finally comes clean: SecurID is compromised

RSA Security will replace almost every one of the 40 million SecurID tokens ...

PETER BRIGHT - 6/6/2011, 7:49 PM



linode

Security Investigation Retrospective

February 19, 2016 3:30 pm

After examining the image from our July investigation, we discovered software capable of generating TOTP codes if provided a TOTP key. We found software implementing the decryption method we use to secure TOTP keys, along with the secret key we use to encrypt them. We also found commands in the bash history

Alternatives

Requires dedicated hardware

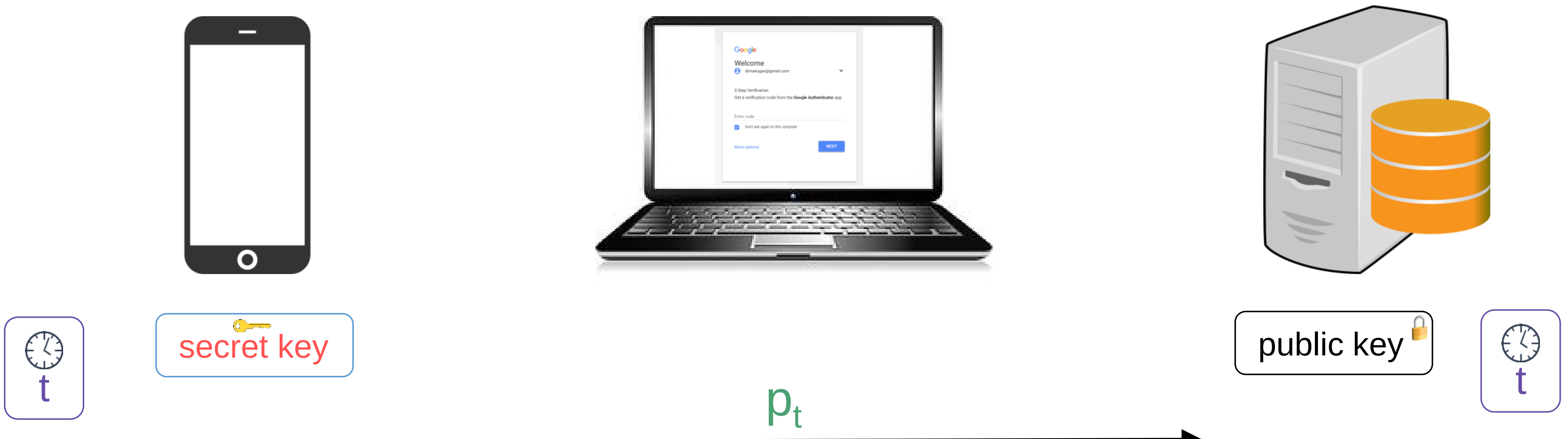
Requires online connection during login

This work: T/Key

- Drop-in replacement for TOTP
- **Store no secrets on the server**

- Additional contributions:
  - Give a new security analysis of hash chains (iterated hash functions)
  - A hash-chain traversal optimization for resource-constrained 2FA devices

Strawman: sign the time



$p_t = \mathrm{sign}_{sk}(t)$ → $\mathrm{verify}_{pk}(t; p_t)$

- 128-bit security requires 512-bit-long signatures
- Even 384-bit-long signatures require 77 Base32 characters
  - compare with 6 digit TOTP codes

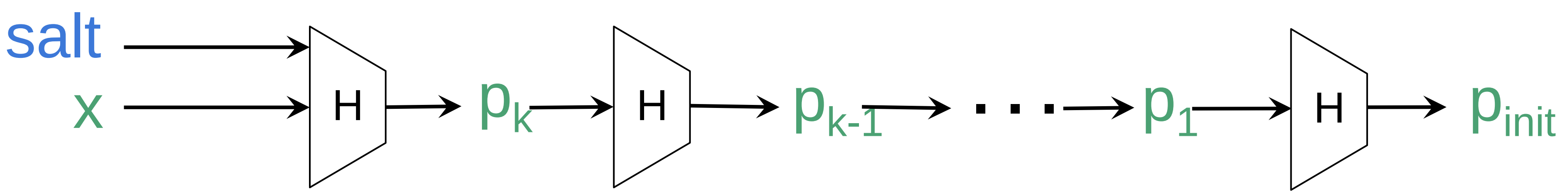
The length challenge

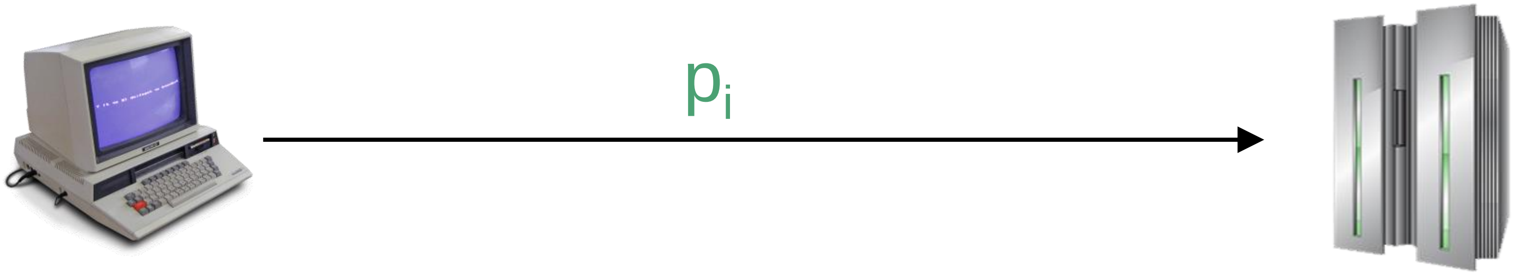
- OTP must not be too long for the user to enter
- OTP must be at least as long as the security parameter
  - Why? An attacker who steals the verifier from the server can do a brute force attack against a future time step
- Challenge: how can we “squeeze” the maximal security from a given OTP length?
  - Ideally, we would want that any attack on scheme with OTP-length n, would require time T as close to 2^n as possible

S/Key [Lam81]

- Hash-chain-based OTP scheme

~~80: SORT ARE NIBS SEAR PUT AJAR~~
~~81: FIB DRAW BRIG SCAN IRK NOAH~~
~~82: ABEL HOME HOP BODE DELL PRY~~
~~83: SHE LOCK IRK LOAD WAS BOCK~~
84: MODE MANY BEET LAB FROM SALE
85: LULU SUNK CRAM SLY SUCH SOOT
86: MUTE HUH VAIL FOOT CULT ALIA
87: BOOM COCA SAUL CREW NINA LENT





$p_{i-1} \stackrel{?}{=} H(p_i)$

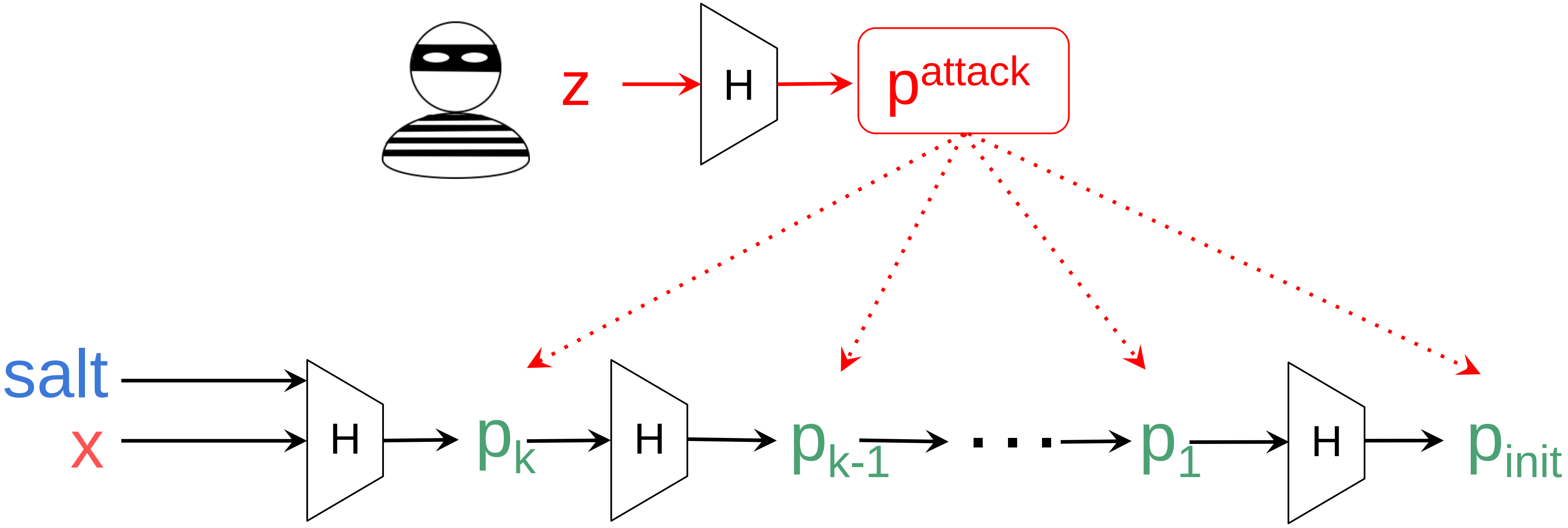
- Passwords are 64-bit long
- Stored on a piece of paper as 6-word phrases

S/Key: the problems

- OTPs are not time based
  - ⇒ easier to phish
  - ⇒ multiple servers must coordinate to avoid replay attacks
- Salt is used in the first iteration only
  - ⇒ susceptible to preprocessing attacks
- Intended to support only a small number of logins (~100)
- The effect of the length of the chain on security is unclear
  - ⇒ what password length should we use?

S/Key security

- Finding a preimage of the k^{th} iterate is k times easier [HN07]
- A million-long chain ⇒ million-times-faster preimage attack
  ⇒ requires two additional words in the passphrase



T/Key: modernizing S/Key



x secret

salt

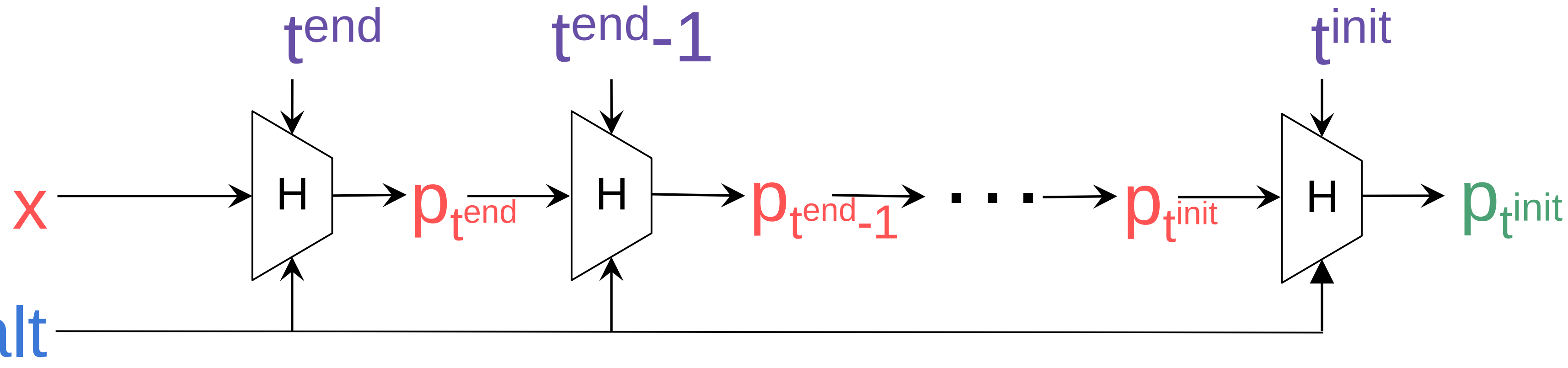
t_{end} - expiration time (now + 4 years)

p_{init} initial verifier

salt

t_{init} - initial time

$$p_{init} = \mathbf{H}(salt || t^{init} || \mathbf{H}(salt || t+1 || \mathbf{H}(\ldots \mathbf{H}(salt || t^{end} || x)\ldots)))$$



T/Key: modernizing S/Key

x : secret

salt

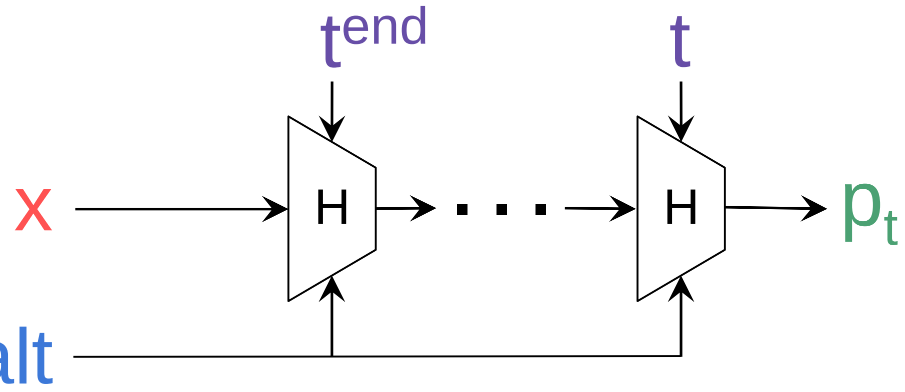
t_{end} : expiration time (now + 4 years)

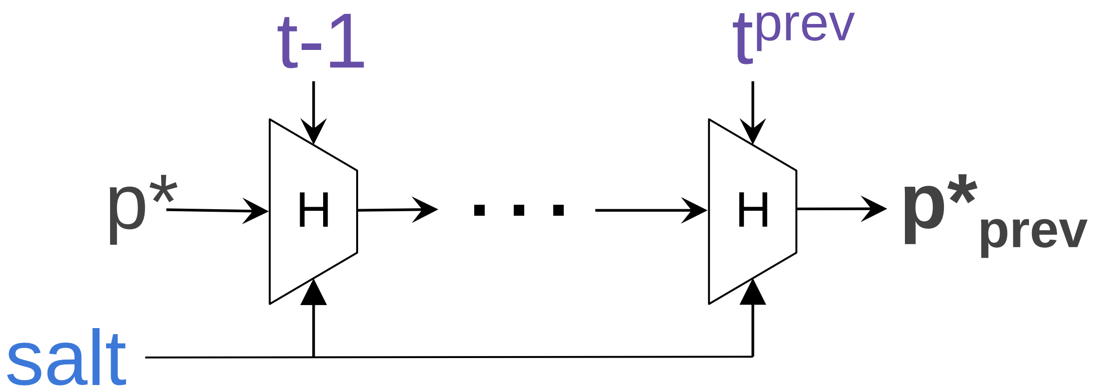
p_{prev} : previous password

salt

t_{prev} : previous auth time

t





T/Key

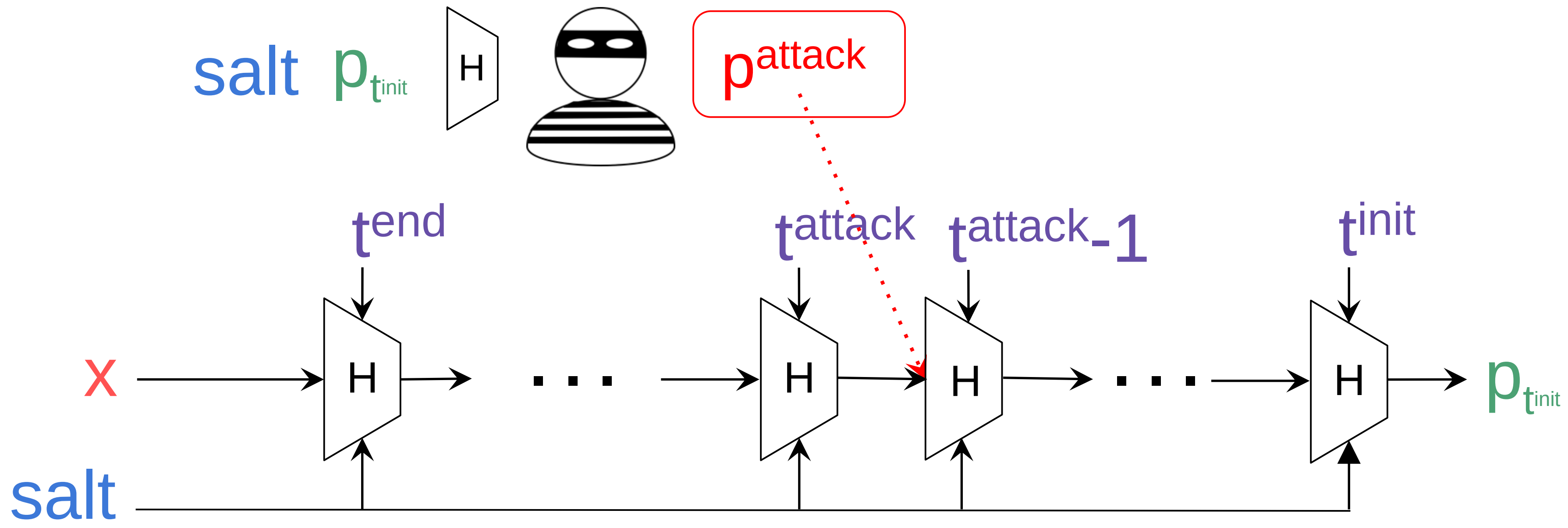
4 years @ 30 second intervals → chain of length 4 million

- **Security**: how does the security degrade with the chain length?
- **Performance**: optimize chain traversal to minimize OTP generation time.

T/Key Security

T/Key Security

- The security game:
  - Attacker hacks into the server or has previously phished an OTP + salt
  - Attacker wants to generate a new valid OTP
- Guessing OTP requires inverting a segment of a hash chain



T/Key Security (in the Random Oracle Model)

Theorem: Consider T/Key with **OTP length n** and **hash chain length k**. Let A be an adversary that makes at most **T random oracle queries**. Then,

$$\Pr[A\ wins] \leq \frac{2T + 2k + 1}{2^n}.$$

The security loss is **additive,** rather than **multiplicative** as in S/Key, for which there exists an attack with

$$\Pr[A\ wins] \geq \Omega\left(\frac{Tk}{2^n}\right)$$

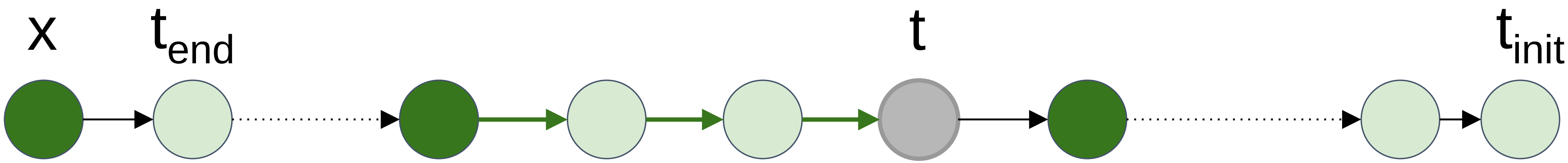
for $k \leq 2^{\frac{n}{2}}, 2k \leq T \leq \frac{2^n}{k}$.

Can reduce the passphrase by two words

Performance

Optimizing OTP-generation time

- Generating an OTP requires traversing a long hash chain
  - Directly translates to login latency
- Approach: store some precomputed checkpoints



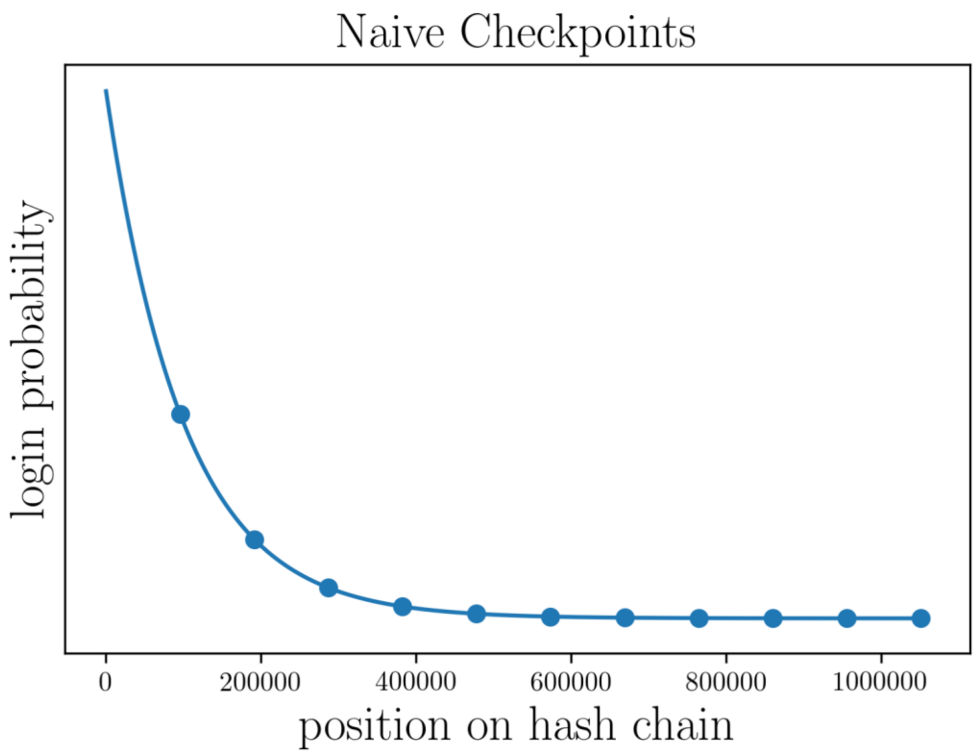
- Previous schemes optimize for **sequential access** [CJ03]
- OTP logins result in **access with gaps**

Our model

- We differentiate between:
  - Query time – time to compute the required OTP
  - Postprocessing time – time to reposition the checkpoints
- Only the query time affects login latency

Optimizing OTP-generation time

- Optimizing for the worst case ⇒ place checkpoints at equal distances
- But we can improve average-case performance, if we know the distribution of login intervals



Optimizing OTP-generation time

$d(t)$ - probability distribution of login intervals

Find checkpoint positions $c_1,\ldots,c_q$ to minimize:

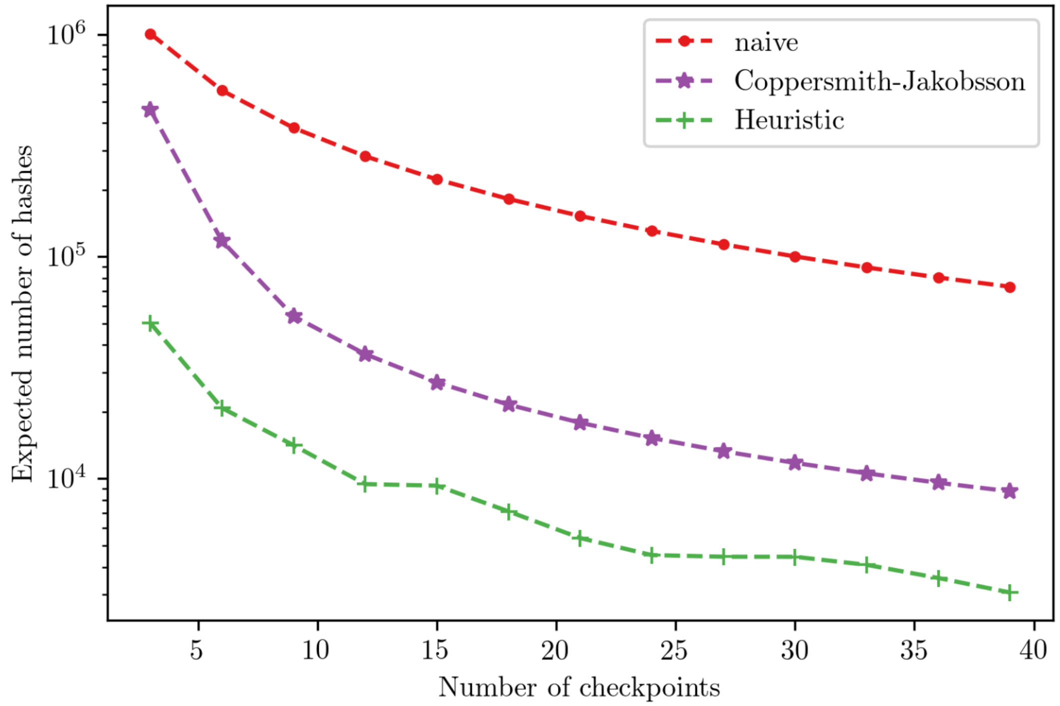
$$\mathbb{E}[cost] = \sum_{i=0}^{q-1} \sum_{t=c_i+1}^{c_{i+1}} (c_{i+1} - t)d(t) = 0$$

Instead of solving multi-variate optimization, apply the following heuristics:

1. Start with full interval $[0, \ell]$
2. Find optimal position within the interval **for one checkpoint**
3. Recurse

Optimizing OTP-generation time

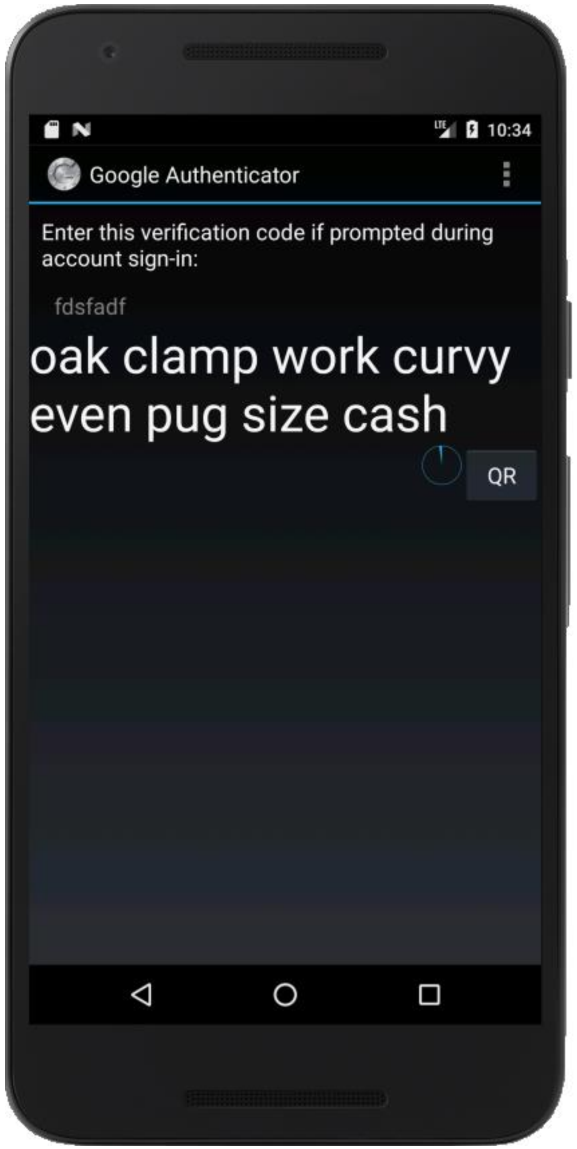
- Model logins as a Poisson process (exponential distribution) [BBD13]



Chain length 4 million (4 years when using 30-second time slots)
Logins modelled as Poisson process with mean 40320 (two weeks)

Implementation

- Extended Google Authenticator
  - Android app for client
  - Linux pam module for server
- 80-bit security - 8-word OTPs
- 128-bit security - 12-word OTPs
  - Can also encode as QR codes



Evaluation

- Client — mobile phone, Server — laptop

Auth. Period	Mean Time Between Logins	Setup Time (seconds)	Password Generation Time (seconds)		Verification Time (seconds)
			average case	worst case	
1 year	1 week	7.5	0.3	0.6	0.4
2 years	2 weeks	14	0.5	0.9	0.8
4 years	1 month	28	0.8	1.6	1.6

Open problems

- Construct an OTP scheme with sub-linear traversal
  - Can we use some tree-like construction?
- Can we reduce OTP length by having a different security level for online attacks and attacks on the server?

Summary

- 2FA scheme without secrets on the server
- Hash chains give a much shorter alternative compared to signatures
- New bounds on the security of hash chains
- Non sequential traversal of hash chains

Thank you for listening!

References

[BBD13] Blocki, Blum, and Datta. Naturally Rehearsing Passwords. ASIACRYPT.

[CJ03] Coppersmith and Jakobsson. Almost Optimal Hash Sequence Traversal. Financial Cryptography.

[DGK17] Dodis, Guo, and Katz. Fixing Cracks in the Concrete: Random Oracles with Auxiliary Input, Revisited. EUROCRYPT.

[DTT10] De, Trevisan and Tulsiani. Time Space Tradeoffs for Attacks against One-Way Functions and PRGs. CRYPTO.

[FN91] Fiat and Naor. Rigorous Time/Space Tradeoffs for Inverting Functions. STOC.

[GT00] Gennaro and Trevisan. Lower Bounds on the Efficiency of Generic Cryptographic Constructions. FOCS.

[Hel80] Hellman. A cryptanalytic time-memory trade-off. IEEE transactions on Information Theory.

[HN07] Håstad and Näslund. Practical Construction and Analysis of Pseudo-Randomness Primitives. J. Cryptol.

[Lam81] Password Authentication with Insecure Communication. Comm ACM.

[LM95] Leighton and Silvio Micali. Large provably fast and secure digital signature schemes based on secure hash functions. . US Patent 5,432,852.

[Oec03] Making a Faster Cryptanalytic Time-Memory Trade-Off. CRYPTO.

[MMPR11] M'Raihi, Machani, Pei, and Rydell. TOTP: Time-Based OneTime Password Algorithm. RFC 6238.

[SJSN14] Shirvanian, Jarecki, Saxena, and Nathan. Two-Factor Authentication Resilient to Server Compromise Using Mix Bandwidth Devices. NDSS.

[Yao90] Yao. Coherent Functions and Program Checkers. STOC.