Hash-Based Authentication Revisited in the Age of High-Performance Computers

Niclas Hedam, Jakob Mollerup, Pınar Tözün

IT University of Copenhagen

Properties of Hashing Functions

- Given M, finding h(M) should be *easy.*
- *Pre-image resistance:*
  Given d, finding M where h(M) = d should be *infeasible.*
- *Second pre-image resistance:*
  Given M, finding M' with h(M) = h(M') should be *infeasible*.
- Collision Resistance:
  Finding M and M' with h(M) = h(M') should be *infeasible.*

Hashing Functions

- Hashing functions should be fast enough for it’s intended application.
- .. but slow enough that it is infeasible to brute-force.

Hashing Functions – Cost Parameters

- MD5 uses an inner loop with 1000 iterations to slow down.
- Hashing functions with a static cost parameter will have to be replaced over time.
- Modern hashing functions, like bcrypt, allow dynamic adjustment of the cost parameter.

Applications

- Authentication schemes/protocols
- Version control (git uses SHA-1)
  - Moving to SHA-256 since SHA-1 are vulnerable
- Indexing in databases

Authentication in practice …

- Many major websites allow passwords that are less than or equal to 8 characters.
- Some major websites allow purely alphanumeric passwords, including Google, LinkedIn and Amazon.
- What if the database of one of these websites were breached or leaked?

Service	Min. Password Length
Wikipedia	1
Netflix	4
Facebook	6
Reddit	6
Amazon	6
LinkedIn	6
Instagram	6
Ebay	6
Yahoo	8
Google	8
Microsoft	8

Our goal

- Given M, finding h(M) should be *easy.*

We want to challenge the 2nd property of hashing functions.

- *Pre-image resistance:*
  Given d, finding M where h(M) = d should be *infeasible.*

- *Second pre-image resistance:*
  Given M, finding M' with h(M) = h(M') should be *infeasible*.

- Collision Resistance
  Finding M and M' with h(M) = h(M') should be *infeasible.*

Outline

- Background on Hashing & Motivation
- Methodology & setup
- Results
- Discussion: Practical security of hash-based authentication

Methodology

- Testing via brute-force attack using hardware available to us.
- Infeasible for long, but feasible for relatively short passwords.
- Benchmark:

- Tested widely-used hashing algorithms: MD5, SHA-1, NTLM.

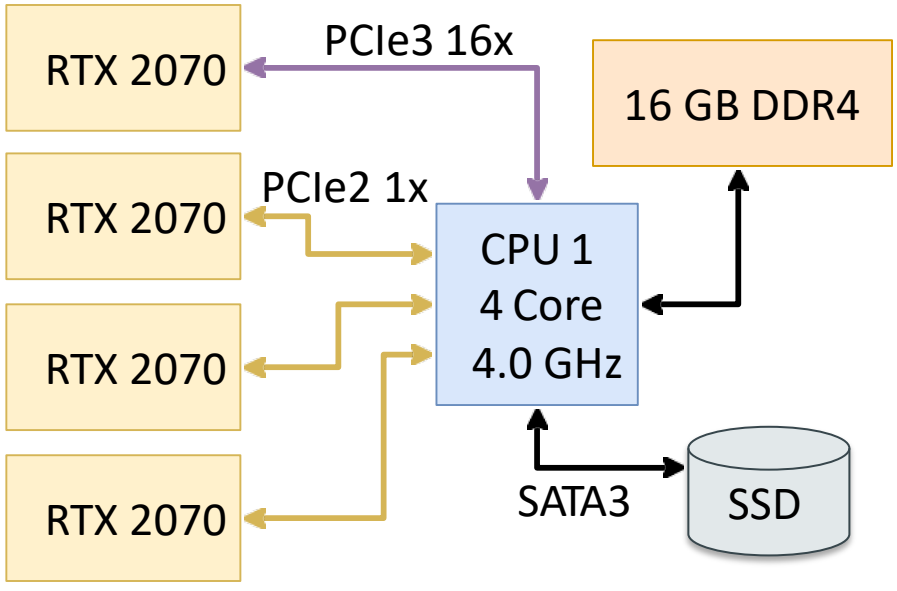
Setup

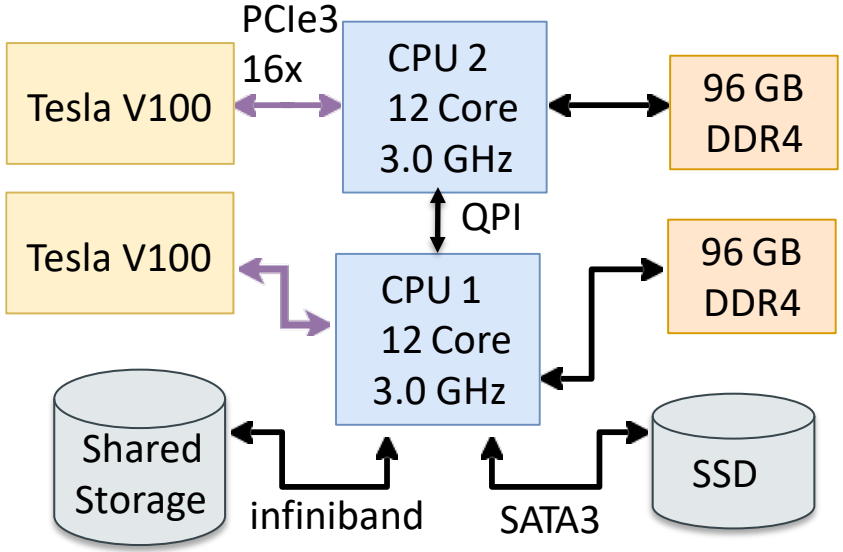
- Measured hashing throughput on four different GPUs
  - Tesla V100 (Server, Enterprise)
  - RTX 2070 (Desktop, High-end Home)
  - GTX 1070 (Desktop, Home)
  - GTX 770 (Desktop, Home)

- On RTX 2070 we also tested
  - PCIe 2.0 vs PCIe 3.0
  - OpenCL vs CUDA

- Throughput measured in millions of hashes per second = MH/s

Hardware


RTX 2070
RTX 2070
RTX 2070
RTX 2070
PCIe3 16x
PCIe2 1x
16 GB DDR4
CPU 1
4 Core
4.0 GHz
SATA3
SSD

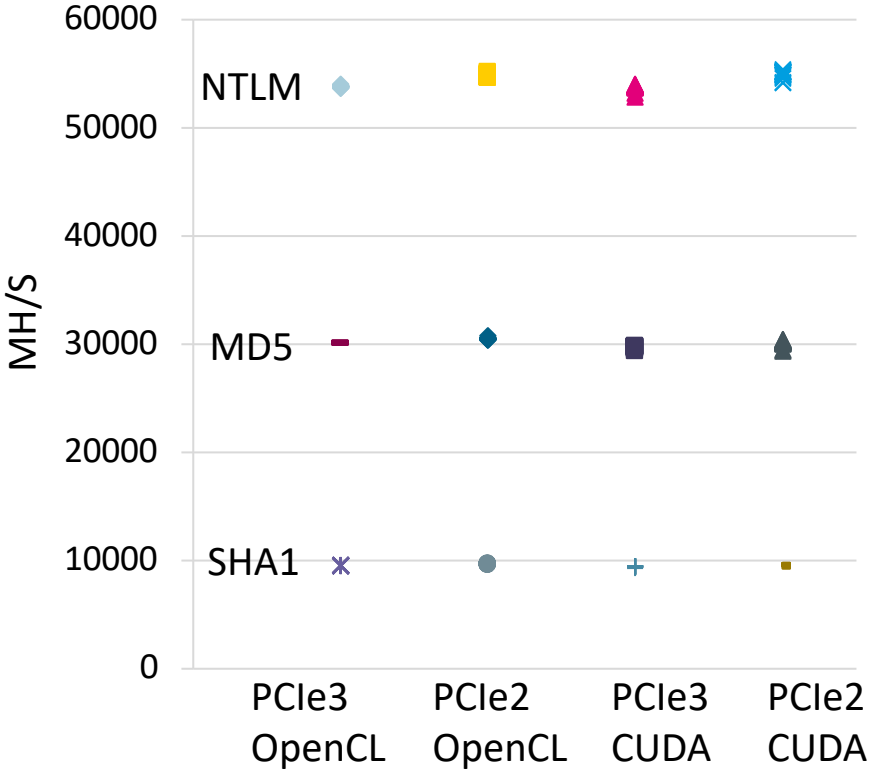

Tesla V100
Tesla V100
PCIe3
16x
CPU 2
12 Core
3.0 GHz
96 GB
DDR4
QPI
CPU 1
12 Core
3.0 GHz
96 GB
DDR4
Shared
Storage
infiniband
SATA3
SSD

Outline

- Background on Hashing & Motivation
- Methodology & setup
- Results
- Discussion: Practical security of hash-based authentication

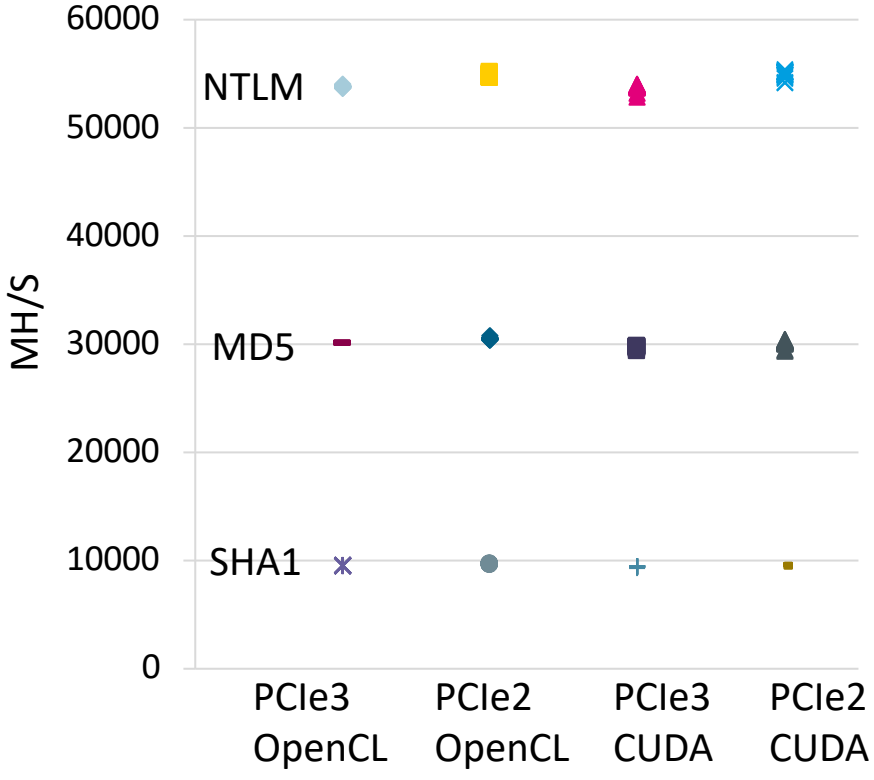
CUDA vs OpenCL

- OpenCL is, on average, ~1% faster for MD5 and SHA-1.
- But no significant differences overall between OpenCL and CUDA.

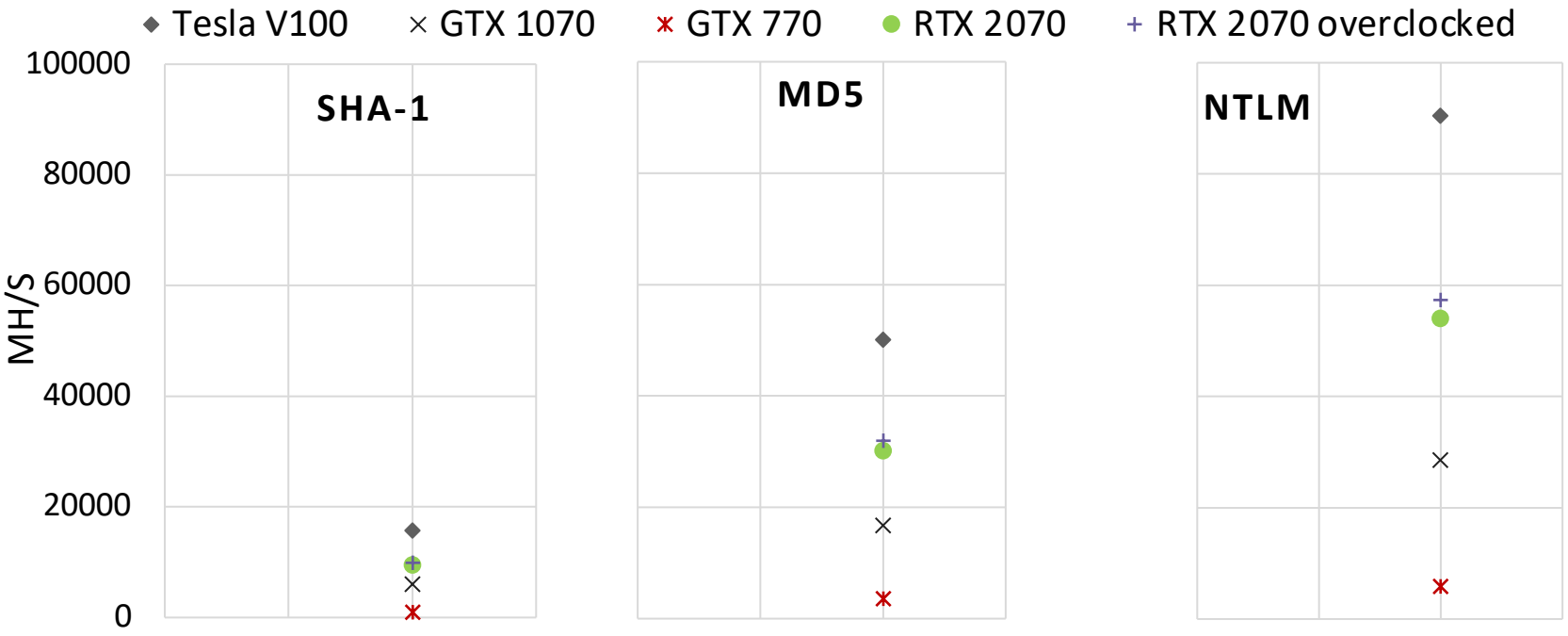


PCIe 2.0 vs. PCIe 3.0

- PCIe 2.0 is faster than PCIe 3.0 for all hashing algorithms.
- But no huge differences, at most ~2%.
- Hashing is not data-intensive enough to utilize/exploit PCIe 3.0 well.

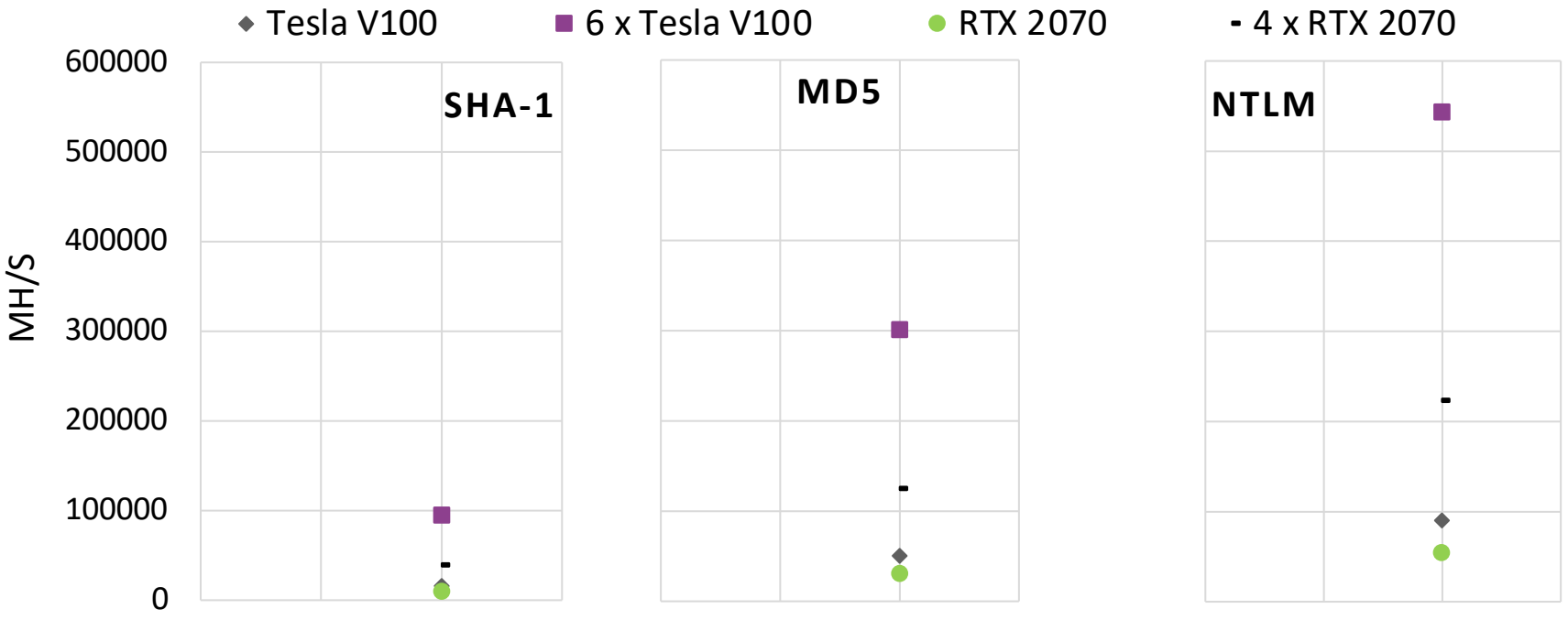


All GPUs – single node



- All results (except one) are in the magnitude of billion hashes per second.
- V100 is an order of magnitude more expensive than RTX 2070, but only 1.5 to 2 times faster.

Multi-GPU



- The performance order of the GPUs are as expected.
- Results show that hashing is embarrassingly parallel.

Heating issues?

- During sensitivity testing of the RTX 2070 we discovered a correlation between waiting-time between runs and the performance.

- Many GPUs induce artificial slowdown when becoming too hot to protect from permanent damage.

- Tesla V100 slows to 50% at 87℃ and shuts down at 90℃.

- Waiting 10 minutes between each run temporarily countered the issue.

Outline

- Background on Hashing & Motivation
- Methodology & setup
- Results
- Discussion: Practical security of hash-based authentication

Let’s get back to real-world

- Entropy is the maximum number of hashes needed to break any alphanumeric password of minimum length.
- We demonstrated that even cheaper GPUs can hash in the magnitude of billions hashes per second.

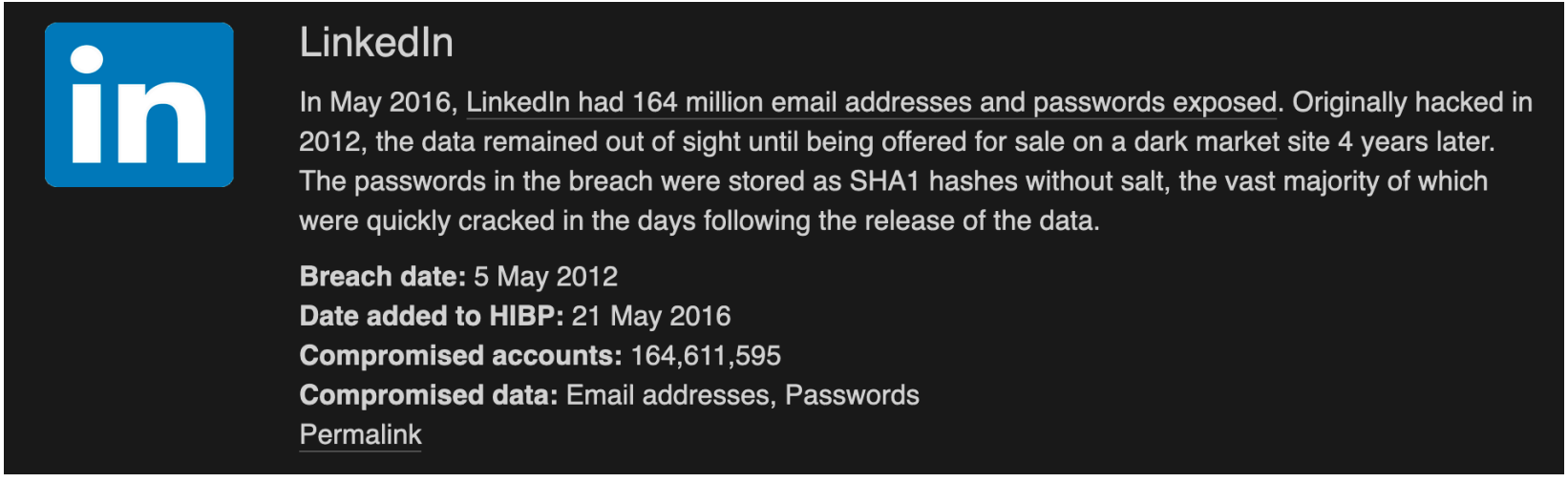
Service	Length	Entropy
Wikipedia	1	62
Netflix	4	14,776,336
Facebook	6	56,800,235,584
Reddit	6	56,800,235,584
Amazon	6	56,800,235,584
LinkedIn	6	56,800,235,584
Instagram	6	56,800,235,584
Ebay	6	56,800,235,584
Yahoo	8	218,340,105,584,896
Google	8	218,340,105,584,896
Microsoft	8	218,340,105,584,896

Time to break passwords

GPU - Length	6	8	10	12
Tesla V100, MD5	1 sec	1 hour	194 days	2042 years
Tesla V100, SHA-1	4 sec	4 hours	616 days	6481 years
Tesla V100, NTLM	1 sec	40 min	107 days	1129 years
RTX 2070, MD5	2 sec	2 hours	323 days	3401 years
RTX 2070, SHA-1	6 sec	6 hours	1024 days	10774 years
RTX 2070, NTLM	1 sec	1 hour	180 days	1892 years

- Recent famous breaches have shown a mean password length of 9.
- Our high-end home GPU could breach alphanumeric passwords of length 9 in a little over two weeks.

History



LinkedIn

In May 2016, LinkedIn had 164 million email addresses and passwords exposed. Originally hacked in 2012, the data remained out of sight until being offered for sale on a dark market site 4 years later. The passwords in the breach were stored as SHA1 hashes without salt, the vast majority of which were quickly cracked in the days following the release of the data.

Breach date: 5 May 2012
Date added to HIBP: 21 May 2016
Compromised accounts: 164,611,595
Compromised data: Email addresses, Passwords
Permalink

- LinkedIn lost 164 million email and password pairs in 2012.
- The passwords were hashed with SHA-1 and unsalted.

Implications

- Personal passwords leaked for everyone to see.
- Reuse of passwords can compromise many more accounts.
- **Passwords are used for (alleged) blackmail!**

```
Hello!

I'm a member of an international hacker group.

As you could probably have guessed, your account niclas@[redacted] was hacked.
On moment of infection niclas@[redacted] was this password: [redacted]

Within a period from December 21, 2019 to May 15, 2020, you were infected by
the virus we've created, through an adult website you've visited.
So far, we have access to your messages, soal media accounts, and messengers.
Moreover, we've gotten full dumps of these data.

We are aware of your little and big secrets...yeah, you do have that. We saw
and recorded your doings on porn websites.
Your tastes are so weird, you know..

But the key thing is that sometimes we recorded you with your webcam, syncing
the recordings with what you watched!
You did terrible things with your body...
I think you are not interested show this video to your friends, relatives, and
your intimate one...

Transfer $966 to my Bitcoin wallet: 19bWrB86z9ojTFnytNRT2k4jcGwTcmRysN
I guarantee that after that, we'll erase all your "data"!
```

Conclusion

- Password requirements of major websites are insufficient, as modern hardware can easily crack weak and moderate passwords.
- While MD5, SHA-1 and NTLM are theoretically secure, they are not practically secure when users have weak, but allowed passwords.
- A thorough survey and analysis of today's authentication methods for data-intensive systems and applications would be very valuable for our community.

Conclusion

- Password requirements of major websites are insufficient, as modern hardware can easily crack weak and moderate passwords.
- While MD5, SHA-1 and NTLM are theoretically secure, they are not practically secure when users have weak, but allowed passwords.
- A thorough survey and analysis of today's authentication methods for data-intensive systems and applications would be very valuable for our community.

Thank you!