BTI 4202: Security and Trust in Distributed Systems

Christian Grothoff

Berner Fachhochschule

13.5.2022

Risk Analysis

Learning objectives

- Fallacies of distributed computing
- Boyd's theorem
- CAP Theorem
- Zooko's Triangle
- Self stabilization
- Attacks and defenses
- Distributed Hash Tables
  - CAN
  - Chord
  - Kademlia
- Advanced Cryptographic Primitives
- Secure Multiparty Computation
- Bonus: Hardware

Part I: Security in Distributed Systems

The 8 Fallacies of Distributed Computing[1]

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology does not change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

[1] According to Peter Deutsch and James Gosling

Limits on authentication

Theorem (Boyd's Theorem I)

"Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states."

Theorem (Boyd's Theorem II)

"Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other."

Solution space: Zfone Authentication (ZRTP) [17]

Idea: combine human interaction proof and baby duck approach:

- A and B perform Diffie-Hellman exchange
- Keying material from previous sessions is used (duckling)
- Short Authentication String (SAS) is generated (hash of DH numbers)
- Both users read the SAS to each other, recognize voice

$\Rightarrow$ ZRTP foils standard man-in-the-middle attack.

CAP Theorem [9]

No distributed system can be *consistent*, *available* and *partition tolerant* at the same time.

- Consistency: A *read* sees the changes made by all previous *writes*
- Availability: *Reads* and *writes* always succeed
- Partition tolerance: The system operates even when network connectivity between components is broken

Blockchain Trilemma

Blockchains claim to achieve three properties:

- Decentralization: there are many participants, and each participant only needs to have a small amount of resources, say $O(c)$
- Scalability: the system scales to $O(n) > O(c)$ transactions
- Security: the system is secure against attackers with $O(n)$ resources

The Blockchain trilemma is that one can only have two of the three.

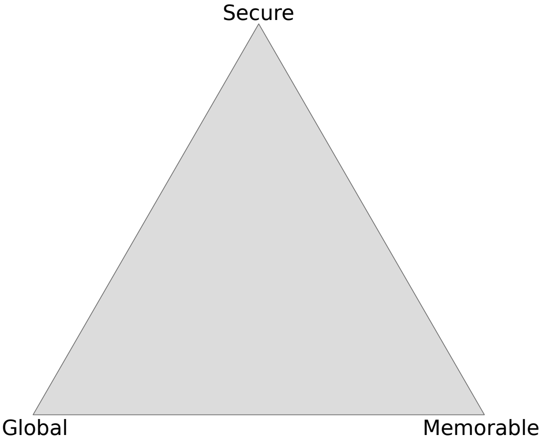
Ryge's Triangle

Ryge's Triangle postulates three key management goals for a system associating cryptographic keys with addresses or names:

- Non-interactive: the system should require no user interface
- Flexible: addresses/names can be re-used by other participants
- Secure: the system is secure against active attackers

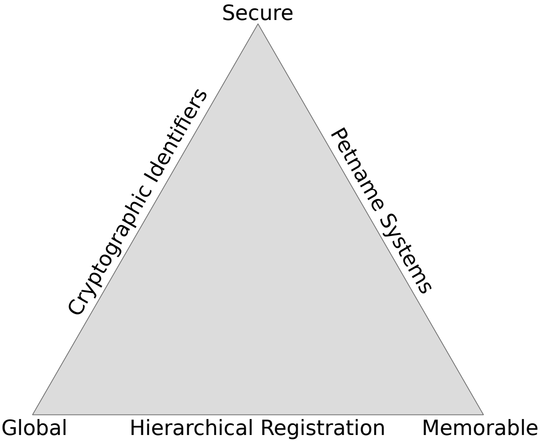
Ryge's triangle says that one can only have two of the three.

Zooko's Triangle



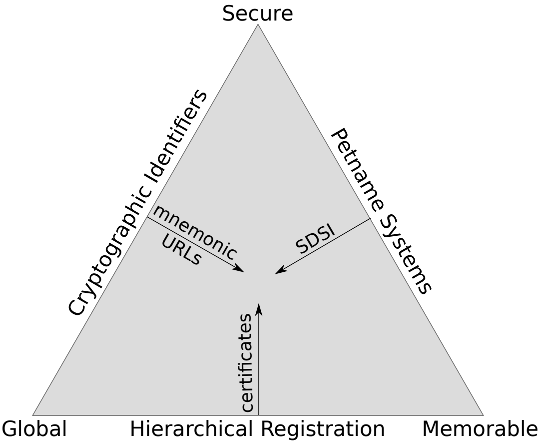
A name system can only fulfill **two**!

Zooko's Triangle



DNS, ".onion" IDs and `/etc/hosts/` are representative designs.

Zooko's Triangle



DNSSEC security is limited (adversary model!)

Self stabilization (Dijkstra 1974)

- A system is self-stabilizing, if starting from any state, it is guaranteed that the system will eventually reach a correct state (convergence).
- Given that the system is in a correct state, it is guaranteed to stay in a correct state, provided that no fault happens (closure).
- Self-stabilization enables a distributed algorithm to recover from a transient fault **regardless of its nature**.

Example: Spanning-tree Protocol from Networking!

Sybil Attack

Background:

- Ancient Greece: Sybils were prophetesses that prophesized under the devine influence of a deity. Note: At the time of prophecy not the person but a god was speaking through the lips of the sybil.
- 1973: Flora Rheta Schreiber published a book "Sybil" about a woman with 16 separate personalities.

Sybil Attack

Background:

- Ancient Greece: Sybils were prophetesses that prophesized under the devine influence of a deity. Note: At the time of prophecy not the person but a god was speaking through the lips of the sybil.
- 1973: Flora Rheta Schreiber published a book “Sybil” about a woman with 16 separate personalities.

The Sybil Attack [8]:

- Insert a node multiple times into a network, each time with a different identity
- Position a node for next step on attack:
  - Attack connectivity of the network
  - Attack replica set
  - In case of majority votes, be the majority.

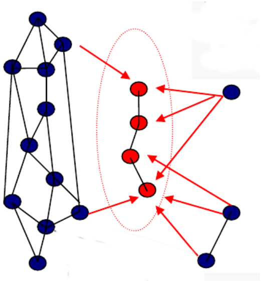
Defenses against Sybil Attacks

- Use authentication with trusted party that limits identity creation
- Use "external" identities (IP address, MAC, e-mail)
- Use "expensive" identities (solve computational puzzles, require payment)

Douceur: Without trusted authority to certify identities, no realistic approach exists to completely stop the Sybil attack.

Eclipse Attack: Goal

- Separate a node or group of nodes from the rest of the network
- isolate peers (DoS, surveillance) or isolate data (censorship)



Eclipse Attack: Techniques

- Use Sybil attack to increase number of malicious nodes
- Take over routing tables, peer discovery

$\Rightarrow$ Details depend on overlay structure

Eclipse Attack: Defenses

- Large number of connections
- Replication
- Diverse neighbour selection (different IP subnets, geographic locations)
- Aggressive discovery ("continuous" bootstrap)
- Audit neighbour behaviour (if possible)
- Prefer long-lived connections / old peers

Poisoning Attacks

Nodes provide false information:

- wrong routing tables
- wrong meta data
- wrong performance measurements

Timing Attacks [14]

Nodes can:

- measure latency to determine origin of data
- delay messages
- send messages using particular timing patterns to aid correlation
- include wrong timestamps (or just have the wrong time set...)

Break

Part II: Distributed Hash Tables

Distributed Hash Tables (DHTs)

- Distributed **index**
- GET and PUT operations like a hash table
- JOIN and LEAVE operations (internal)
- Trade-off between JOIN/LEAVE and GET/PUT costs
- Typically use exact match on cryptographic hash for lookup
- Typically require overlay to establish particular connections

DHTs: Key Properties

To know a DHT, you must know (at least) its:

- routing table structure
- lookup procedure
- join operation process
- leave operation process

... including expected costs (complexity) for each of these operations.

A trivial DHTs: The Clique

- routing table: hash map of all peers
- lookup: forward to closest peer in routing table
- join: ask initial contact for routing table, copy table, introduce us to all other peers, migrate data we're closest to to us
- leave: send local data to remaining closest peer, disconnect from all peers to remove us from their routing tables

Complexity?

A trivial DHTs: The Circle

- routing table: left and right neighbour in cyclic identifier space
- lookup: forward to closest peer (left or right)
- join: lookup own peer identity to find join position, transfer data from neighbour for keys we are closer to
- leave: ask left and rigt neighbor connect directly, transfer data to respective neighbour

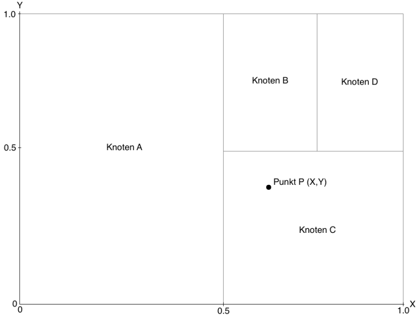
Complexity?

Additional Questions to ask

- Security against Eclipse attack?
- Survivability of DoS attack?
- Maintenance operation cost & required frequency?
- Latency? ($\neq$ number of hops!)
- Data persistence?

Content Addressable Network: CAN

- routing table: neighbours in d-dimensional torus space
- lookup: forward to closest peer
- join: lookup own peer identity to find join position, split quadrant (data areas) with existing peer
- leave: assign quadrant space to neighbour (s)

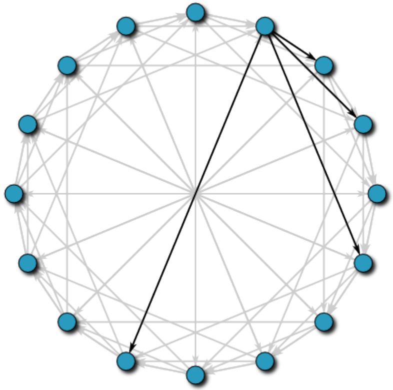


Interesting CAN properties

- CAN can do range queries along $\leq n$ dimensions
- CAN's peers have $2d$ connections (independent of network size)
- CAN routes in $O(d \sqrt[d]{n})$

Chord

- routing table: predecessor in circle and at distance 2^i, plus r successors
- lookup: forward to closest peer (peer ID after key ID)
- join: lookup own peer identity to find join position, use neighbor to establish finger table, migrate data from respective neighbour
- leave: join predecessor with successor, migrate data to respective neighbour, periodic stabilization protocol takes care of finger updates

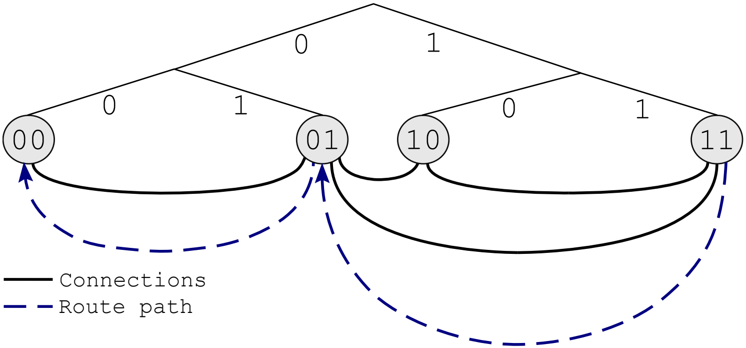


Interesting Chord properties

- Simple design
- $\log_2 n$ routing table size
- $\log_2 n$ lookup cost
- Asymmetric, inflexible routing tables

Kademlia

- routing table: 2^{160} buckets with k peers at XOR distance 2^i
- lookup: iteratively forward to α peers from the "best" bucket, selected by latency
- join: lookup own peer identity, populate table with peers from iteration
- maintenance: when interacting with a peer, add to bucket if not full; if bucket full, check if longest-not-seen peer is live first
- leave: just drop out



Interesting Kademlia properties

- XOR is a symmetric metric: connections are used in both directions
- α replication helps with malicious peers and churn
- Iterative lookup gives initiator much control,
- Lookup helps with routing table maintenance
- Bucket size trade-off between routing speed and table size
- Iterative lookup is a trade-off:
  - good UDP (no connect cost, initiator in control)
  - bad with TCP (very large number of connections)

Part III: Advanced Cryptographic Primitives

Homomorphic Encryption

$$E(x_1 \oplus x_2) = E(x_1) \otimes E(x_2) \tag{1}$$

Multiplicative Homomorphism: RSA & ElGamal

- Unpadded RSA (multiplicative):

$$E(x_1) \cdot E(x_2) = x_1^e x_2^e = E(x_1 \cdot x_2) \tag{2}$$

- ElGamal:

$$\begin{aligned} E(x_1) \cdot E(x_2) &= (g^{r_1}, x_1 \cdot h^{r_1})(g^{r_2}, x_2 \cdot h^{r_2}) && (3) \\ &= (g^{r_1+r_2}), (x_1 \cdot x_2) h^{r_1+r_2}) && (4) \\ &= E(x_1 \cdot x_2) && (5) \end{aligned}$$

Additive Homomorphism: Paillier

$$\mathrm{E}_K(m) := g^m \cdot r^n \mod n^2, \tag{6}$$

$$\mathrm{D}_K(c) := \frac{(c^\lambda \mod n^2) - 1}{n} \cdot \mu \mod n \tag{7}$$

where the public key $K = (n, g)$, m is the plaintext, c the ciphertext, n the product of $p, q \in \mathbb{P}$ of equal length, and $g \in \mathbb{Z}^*_{n^2}$. In Paillier, the private key is (λ, μ), which is computed from p and q as follows:

$$\lambda := \mathrm{lcm}(p - 1, q - 1), \tag{8}$$

$$\mu := \left(\frac{(g^\lambda \mod n^2) - 1}{n}\right)^{-1} \mod n. \tag{9}$$

Paillier offers additive homomorphic public-key encryption, that is:

$$\mathrm{E}_K(a) \otimes \mathrm{E}_K(b) \equiv \mathrm{E}_K(a + b) \tag{10}$$

for any public key K.

Fully homomorphic encryption

Additive:

$$E(A) \oplus E(B) = E(A + B) \tag{11}$$

and multiplicative:

$$E(A) \otimes E(B) = E(A \cdot B) \tag{12}$$

Known cryptosystems: Brakerski-Gentry-Vaikuntanathan (BGV), NTRU, Gentry-Sahai-Waters (GSW).

Pairing-based cryptography

Let G_1, G_2 be two additive cyclic groups of prime order q, and G_T another cyclic group of order q (written multiplicatively). A pairing is an efficiently computable map e:

$$e : G_1 \times G_2 \rightarrow G_T \tag{13}$$

which satisfies $e \neq 1$ and bilinearity:

$$\forall_{a,b \in F_q^*}, \ \forall_{P \in G_1, Q \in G_2} : \ e\left(aP, bQ\right) = e\left(P, Q\right)^{ab} \tag{14}$$

Examples: Weil pairing, Tate pairing.

Hardness assumption

Computational Diffie Hellman:

$$g, g^x, g^y \Rightarrow g^{xy} \tag{15}$$

remains hard on G even given e.

Boneh-Lynn-Sacham (BLS) signatures [7]

Key generation:

Pick random $x \in \mathbb{Z}_q$

Signing:

$\sigma := h^x$ where $h := H(m)$

Verification:

Given public key g^x:

$$e(\sigma, g) = e(h, g^x) \tag{16}$$

Boneh-Lynn-Sacham (BLS) signatures [7]

Key generation:

Pick random $x \in \mathbb{Z}_q$

Signing:

$\sigma := h^x$ where $h := H(m)$

Verification:

Given public key g^x:

$$e(\sigma, g) = e(h, g^x) \tag{16}$$

Why:

$$e(\sigma, g) = e(h, g)^x = e(h, g^x) \tag{17}$$

due to bilinearity.

Fun with BLS

Given signature $\langle \sigma, g^x \rangle$ on message h, we can *blind* the signature and public key g^x:

$$e(\sigma^b, g) = e(h, g)^{xb} = e(h, g^{xb}) \tag{18}$$

Thus σ^b is a valid signature for the *derived* public key $(g^x)^b$ with blinding value $b \in \mathbb{Z}_q$.

Part IV: Secure Multiparty Computation

Secure Multiparty Computation (SMC)

- Alice und Bob haben private Daten a_i and b_i.
- Alice und Bob führen ein Protokoll aus und berechnen gemeinsam $f(a_i, b_i)$.
- Nur einer von beiden lernt das Ergebnis (i.d.R.)

Adversary models

Honest but curious

Dishonest and curious

Secure Multiparty Computation: Scalar Product

- Original idea by Ioannids et al. in 2002 [11] (use: $(a-b)^2 = a^2 - 2ab + b^2$)
- Refined by Amirbekyan et al. in 2007 (corrected math) [3]
- Now providing protocol with practical extensions (negative numbers, small numbers, set intersection).

Preliminaries

- Alice has public key A and input map $m_A : M_A \to \mathbb{Z}$.
- Bob has public key B and input map $m_B : M_B \to \mathbb{Z}$.
- We want to calculate
$$\sum_{i \in M_A \cap M_B} m_A(i) m_B(i) \tag{19}$$
- We first calculate $M = M_A \cap M_B$.
- Define $a_i := m_A(i)$ and $b_i := m_B(i)$ for $i \in M$.
- Let s denote a shared static offset.

Network Protocol

- Alice transmits $\mathsf{E}_A(s + a_i)$ for $i \in M$ to Bob.
- Bob creates two random permutations π and π' over the elements in M, and a random vector r_i for $i \in M$ and sends

$$\begin{aligned} R &:= \mathsf{E}_A(s + a_{\pi(i)}) \otimes \mathsf{E}_A(s - r_{\pi(i)} - b_{\pi(i)}) && (20)\\ &= \mathsf{E}_A(2 \cdot s + a_{\pi(i)} - r_{\pi(i)} - b_{\pi(i)}), && (21)\\ R' &:= \mathsf{E}_A(s + a_{\pi'(i)}) \otimes \mathsf{E}_A(s - r_{\pi'(i)}) && (22)\\ &= \mathsf{E}_A(2 \cdot s + a_{\pi'(i)} - r_{\pi'(i)}), && (23)\\ S &:= \sum (r_i + b_i)^2, && (24)\\ S' &:= \sum r_i^2 && (25) \end{aligned}$$

Decryption (1/3)

Alice decrypts R and R' and computes for $i \in M$:

$$a_{\pi(i)} - b_{\pi(i)} - r_{\pi(i)} = \mathrm{D}_A\left(R\right) - 2 \cdot s, \tag{26}$$

$$a_{\pi'(i)} - r_{\pi'(i)} = \mathrm{D}_A\left(R'\right) - 2 \cdot s, \tag{27}$$

which is used to calculate

$$T := \sum_{i \in M} a_i^2 \tag{28}$$

$$U := -\sum_{i \in M} (a_{\pi(i)} - b_{\pi(i)} - r_{\pi(i)})^2 \tag{29}$$

$$U' := -\sum_{i \in M} (a_{\pi'(i)} - r_{\pi'(i)})^2 \tag{30}$$

Decryption (2/3)

She then computes

$$
\begin{aligned}
P := {} & S + T + U \\
= {} & \sum_{i\in M} (b_i + r_i)^2 + \sum_{i\in M} a_i^2 + \left(- \sum_{i\in M} (a_i - b_i - r_i)^2 \right) \\
= {} & \sum_{i\in M} \left((b_i + r_i)^2 + a_i^2 - (a_i - b_i - r_i)^2 \right) \\
= {} & 2 \cdot \sum_{i\in M} a_i (b_i + r_i). \\
P' := {} & S' + T + U' \\
= {} & \sum_{i\in M} r_i^2 + \sum_{i\in M} a_i^2 + \left(- \sum_{i\in M} (a_i - r_i)^2 \right) \\
= {} & \sum_{i\in M} \left(r_i^2 + a_i^2 - (a_i - r_i)^2 \right) = 2 \cdot \sum_{i\in M} a_i r_i.
\end{aligned}
$$

Decryption (3/3)

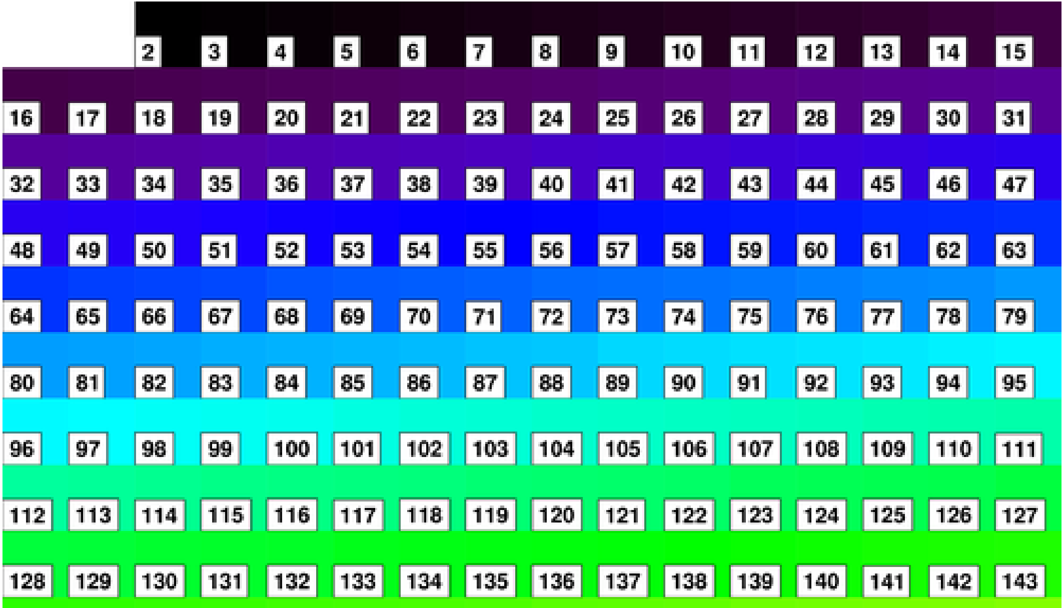
Finally, Alice computes the scalar product using:

$$\frac{P - P'}{2} = \sum_{i \in M} a_i(b_i + r_i) - \sum_{i \in M} a_i r_i = \sum_{i \in M} a_i b_i. \tag{31}$$

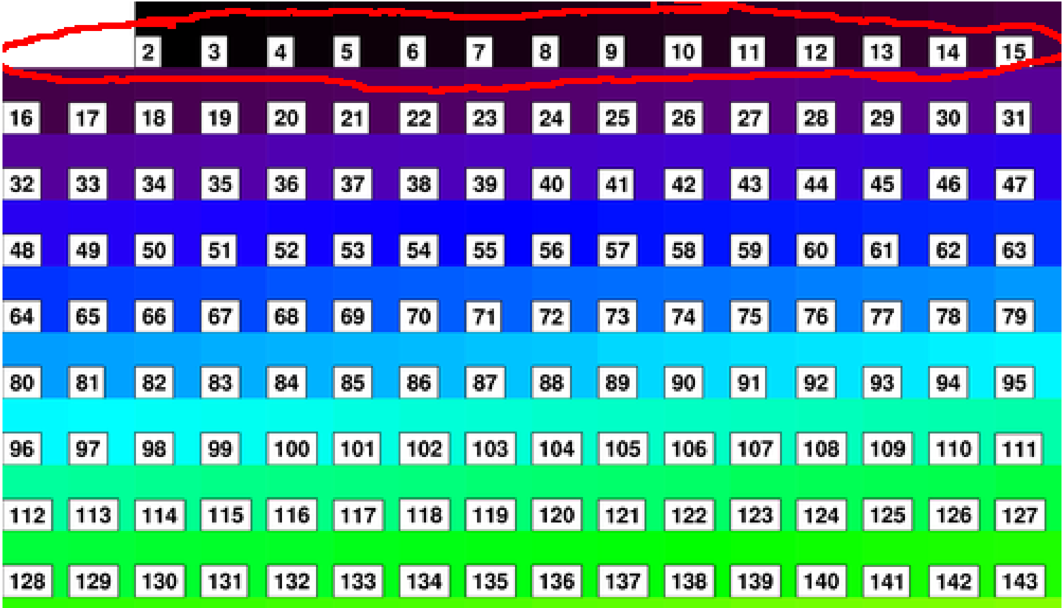
Computing Discrete Logarithms

Who said calculating DLOG was hard?

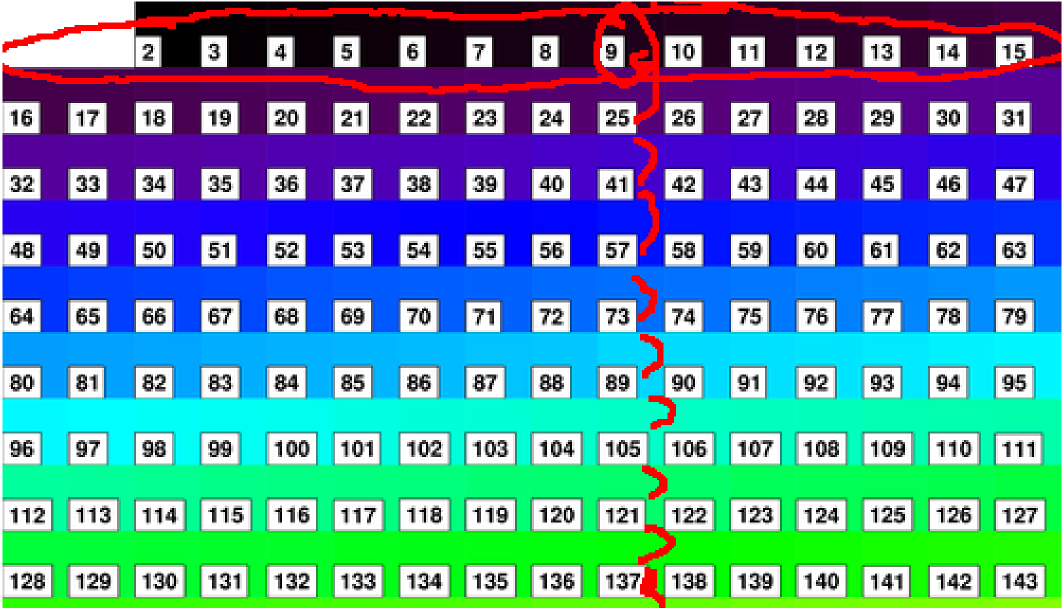
Computing Discrete Logarithms



Baby Steps


2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111
112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

Giant Steps


2 3 4 5 6 7 8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111
112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127
128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143

ECC Version[2]

Let Alice's secret value be a. Alice sends to Bob $(g_i, h_i) = (g^{r_i}, g^{r_i a + a_i})$ with random values r_i for $i \in M$.
Bob answers with:

$$\left(\prod_{i\in M} g_i^{b_i}, \prod_{i\in M} h_i^{b_i}\right) = \left(\prod_{i\in M} g_i^{b_i}, \left(\prod_{i\in M} g_i^{b_i}\right)^a g^{\sum_{i\in M} a_i b_i}\right)$$

Alice can then calculate:

$$\left(\prod_{i\in M} g_i^{b_i}\right)^{-a} \cdot \left(\prod_{i\in M} g_i^{b_i}\right)^a \cdot g^{\sum_{i\in M} a_i b_i} = g^{\sum_{i\in M} a_i b_i}.$$

Assuming $\sum_{i\in M} a_i b_i$ is sufficiently small, then Alice can compute the scalaproduct by solving the DLP.

[2] Joint work with Tanja Lange

Performance Evaluation (AMD Threadripper 1950)

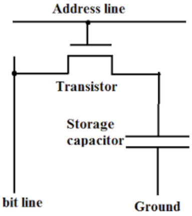
Length	RSA-2048	ECC-2^{20}	ECC-2^{28}
25	4 s	0.1 s	4.2 s
50	8 s	0.1 s	4.3 s
100	10 s	0.2 s	4.3 s
200	19 s	0.2 s	4.3 s
400	35 s	0.3 s	4.3 s
800	74 s	0.4 s	4.5 s
800	1234 kb	65 kb	65 kb

The pre-calculation of ECC-2^{28} is $\times 16$ more expensive than for ECC-2^{20} as the table is set to have size $\sqrt{n}$.

Part V: Hardware (Bonus)

DRAM

DRAM stands for Dynamic Random Access Memory. Each DRAM cell holds one bit of information.

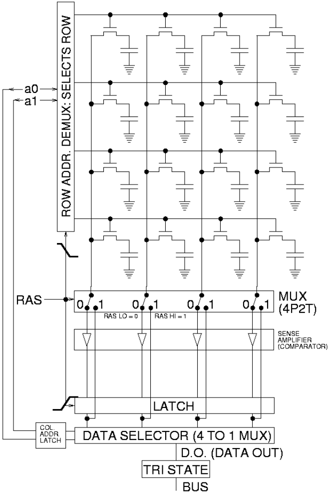


DRAM Cell [6]

Each capacitor in a DRAM cell holds a specific charge. For example 1 Volt representing the bit value of 1 and 0 Volt corresponding to the bit value of 0.

DRAM Modules

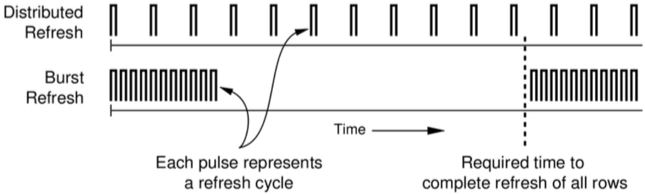
The cells are arranged into rows:



DRAM Cell Matrix [10]

Refresh

As the electric charge decays, it must be periodically **refreshed**. There are two main methods in which the memory is refreshed. One is the Distributed refresh and the other one is a Burst refresh. [5]



DRAM Refreshrate [5]

Rowhammer: Attack idea

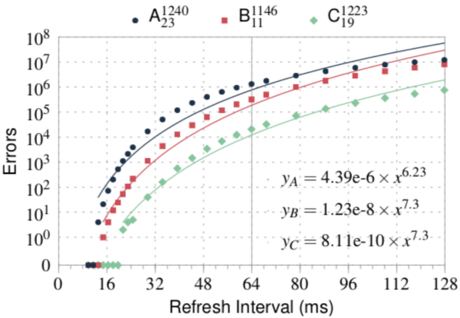
Below is sample code on how to perform the hammering of the Rowhammer attack:

```
attack_loop:
  mov (addr_X), %rax // read the row X
  mov (addr_Y), %rbx // read the row Y
  clflush (addr_X) // flush X from cache
  clflush (addr_Y) // flush Y from cache
  jmp  attack_loop
```

In this code sniped addr_X and addr_Y are our aggressor rows. The values from the aggressor rows are read in an alternating fashion in order to prevent a value to be stored and read from the row buffer rather than actually accessing and reading from the rows in memory.

Without refresh, bits can flip

The plot below shows the direct correlation between the refresh rate and the number of bit flips that were induced.

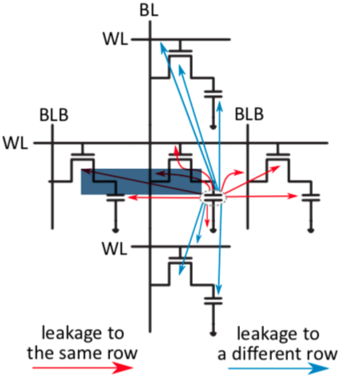


Number of errors occuring when the refresh intervall is increased [12]

As the refresh interval is increased along the x axis the number of errors grows exponentially.

DRAM capacitors are not all the same

Often times the retention rate of a capacitor is not persistent over a reboot. The retention rate of a capacitor is dependent on how much charge their neighbouring cells leak.

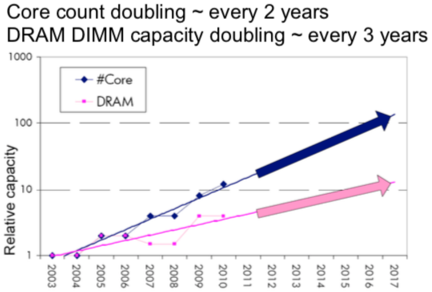


DRAM charge leak [4]

Smaller is not always better

The Rowhammer attack was not feasible before 2010, it was only after the density of cells on a DRAM chip reached a certain threshold that Rowhammer became a feasible attack:

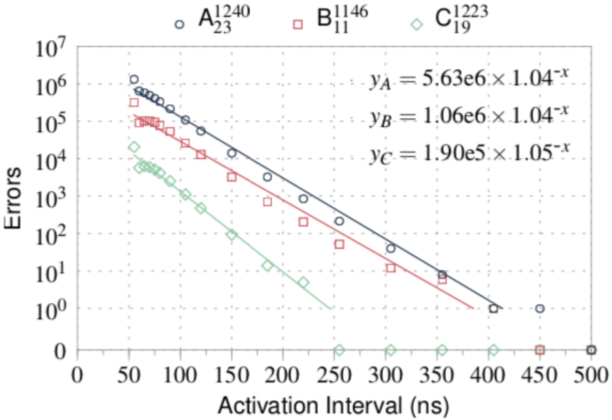
DRAM Density [12]



DRAM Density [2]

Hitting the RAM hard

The graph shows the correlation between the number of row activation of the aggressor row and the number of induced errors (bit flips).



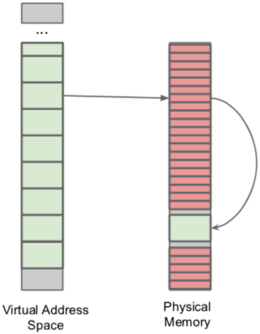
Number of a row activations relative to the amount of induced errors [12]

Privilege escalation with Rowhammer

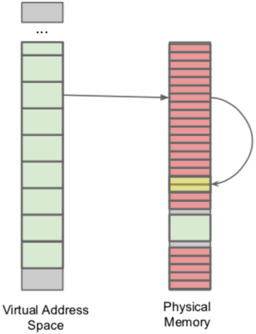
In [1] a 7 step approach is presented to perform a privilege escalation:

1. Allocate a large chunk of memory.
2. Search for a locations prone to flipping.
3. Check if they fall into the "right spot" in a page table entry (PTE) for allowing the exploit.
4. Return that particular area of memory to the operating system.
5. Force the OS to re-use the memory for PTEs by allocating massive quantities of address space.
6. Cause a bit to flip in a PTE resulting in the PTE pointing to a different PTE.
7. Abuse the R/W access to all of the physical memory.

Attack illustration



(a) Location of PTEs [1]



(b) Modified PTE [1]

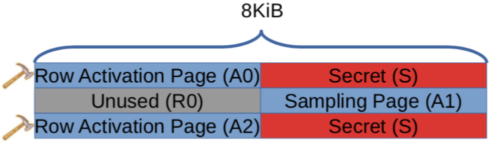
Rowhammer “implementations”

Attacks	Attack Origin				Intended Implication				Methodology											
	Local		Remote						LP				RH						EV	
	Unprivileged Process	Privileged Process	Website	Network	Corrupt EPRO	Corrupt EPUO	Corrupt DPRO	Corrupt DPUO	(A1)Object Spraying	(A2)Forced Padding	(A3)Induced Replacement	(A4)Try and Abort	(Ba1)Specific Instructions	(Ba2)Cache Eviction Set	(Ba3)Uncached Memory	(Bb1)Single-sided	(Bb2)Double-sided	(Bb3)One-location	(C1)Directly Read	(C2)Behaviour Judgement
Flipping bits	✓				✓				✓							✓	✓		✓	
Gain kernel	✓				✓			✓	✓				✓			✓			✓	✓
New Approach	✓				✓			✓	✓				✓			✓			✓	✓
Rowhammer.js	✓		✓					✓	✓					✓		✓				✓
Dedup Est Machina			✓			✓				✓				✓		✓				✓
One Bit Flips		✓			✓						✓		✓				✓		✓	
Flip Feng Shui		✓			✓						✓		✓				✓		✓	
Drammer	✓							✓		✓					✓		✓			✓
Curious Case		✓				✓						✓	✓	✓		✓				✓
Good Go Bad	✓				✓				✓					✓		✓			✓	
SGX-Bomb	✓					✓						✓	✓				✓			✓
Another Flip	✓						✓					✓	✓					✓	✓	
Glitch			✓			✓				✓				✓			✓			✓
Throwhammer				✓				✓		✓					✓		✓			✓
RAMPAGE	✓							✓		✓					✓		✓			✓
Still hammerable	✓							✓		✓			✓			✓				✓
PFA		✓					✓					✓	✓				✓		✓	
Nethammer				✓		✓	✓	✓				✓	✓	✓	✓			✓	✓	✓

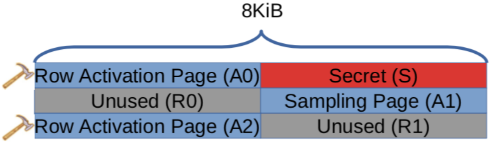
Classification of Rowhammer attacks [15]

Follow-up Research

- ECCploit [16]: Rowhammer for ECC Memory
- RAMbleed [13]: read the content of a memory row without ever accessing it



(a) Double-sided Rambleed. Here, the sampling page (A1) is sandwiched between two copies of S.



(b) Single-sided Rambleed. Here, the sampling page (A1) is neighbored by the secret-containing page (S) on a single side.

RAMBleed page configuration [13]

References I

Exploiting the dram rowhammer bug to gain kernel privileges.
https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn-Exploiting-The-DRAM-Rowhammer-Bug-To-Gain-Kerne pdf.

Understanding and overcoming challenges of dram refresh.
https://people.inf.ethz.ch/omutlu/pub/mutlu˙dram-refresh˙extreme-scale-computing14.pdf.

Artak Amirbekyan and Vladimir Estivill-castro.
A new efficient privacypreserving scalar product protocol.
In *in Proc. of AusDM '07*, pages 209–214.

Nikolaos Athanasios Anagnostopoulos, Stefan Katzenbeisser, John A. Chandy, and Fatemeh Tehranipoor.
An overview of dram-based security primitives.
Cryptography 2018, Volume 2(7), July 2018.

References II

Balasubramanya Bhat and Frank Mueller.
Making dram refresh predictable.
Real-Time Syst., 47(5):430–453, September 2011.

blackandwhitecomputer.
blackandwhitecomputer.
https://blackandwhitecomputer.blogspot.com/2012/03/reading-writing-operation-of-dram.html, March 2012.
Accessed on 2019-05-26.

Dan Boneh, Ben Lynn, and Hovav Shacham.
Short signatures from the weil pairing.
In *Advances in Cryptology – ASIACRYPT '01, LNCS*, pages 514–532. Springer, 2001.

References III

John Douceur.
The Sybil Attack.
In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

Seth Gilbert and Nancy Lynch.
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.
SIGACT News, 33(2):51–59, June 2002.

Glogger.
Dram ece385 illustrative example.
https://upload.wikimedia.org/wikipedia/commons/3/3d/Square_array_of_mosfet_cells_read.png, February 2007.
Accessed on 2019-05-26.

References IV

Ioannis Ioannidis, Ananth Grama, and Mikhail J. Atallah.
A secure protocol for computing dot-products in clustered and distributed environments.
In *31st International Conference on Parallel Processing (ICPP 2002), 20-23 August 2002, Vancouver, BC, Canada*, pages 379–384. IEEE Computer Society, 2002.

Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu.
Flipping bits in memory without accessing them: An experimental study of dram disturbance errors, June 2014.

Andrew Kwong, Daniel Genkin, Daniel Gruss, and Yuval Yarom.
Rambleed: Reading bits in memory without accessing them, 2020.

Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright.
Timing attacks in low-latency mix-based systems.
In *Proceedings of Financial Cryptography (FC '04)*, pages 251–265, February 2004.

References V

Xiaoxuan Lou, Fan Zhang, Zheng Leong Chua, Zhenkai Liang, Yueqiang Cheng, and Yajin Zhou.
Understanding rowhammer attacks through the lens of a unified reference framework, 2019.

Cojocar Lucian, Razavi Kaveh, Giuffrida Cristiano, and Bos Herbert.
Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks, 2019.

Laurianne McLaughlin.
Philip zimmermann on what's next after pgp.
IEEE Security & Privacy, 4(1):10–13, 2006.