VERKLE BASED POST-QUANTUM DIGITAL SIGNATURE WITH LATTICES

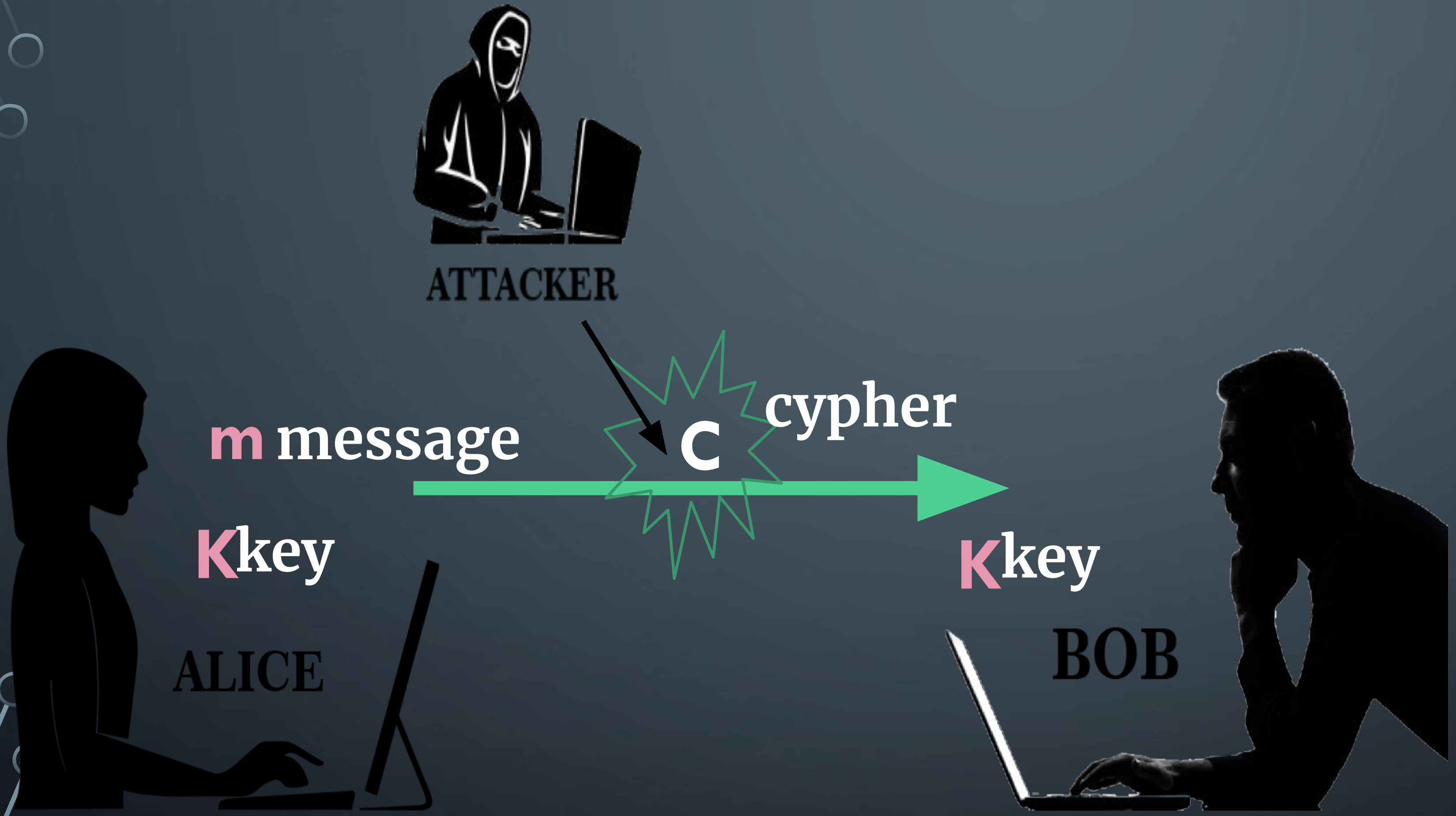
SCIENTIFIC CYBER SECURITY ASSOCIATION

DefCamp
Where Hacking & Security Collide

კავკასიის უნივერსიტეტი
CAUCASUS UNIVERSITY

MAKSIM IAVICH

PRIVATE-KEY CRYPTOGRAPHY



$c := \mathrm{Enc}_k(m)$

Encryption

$\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$

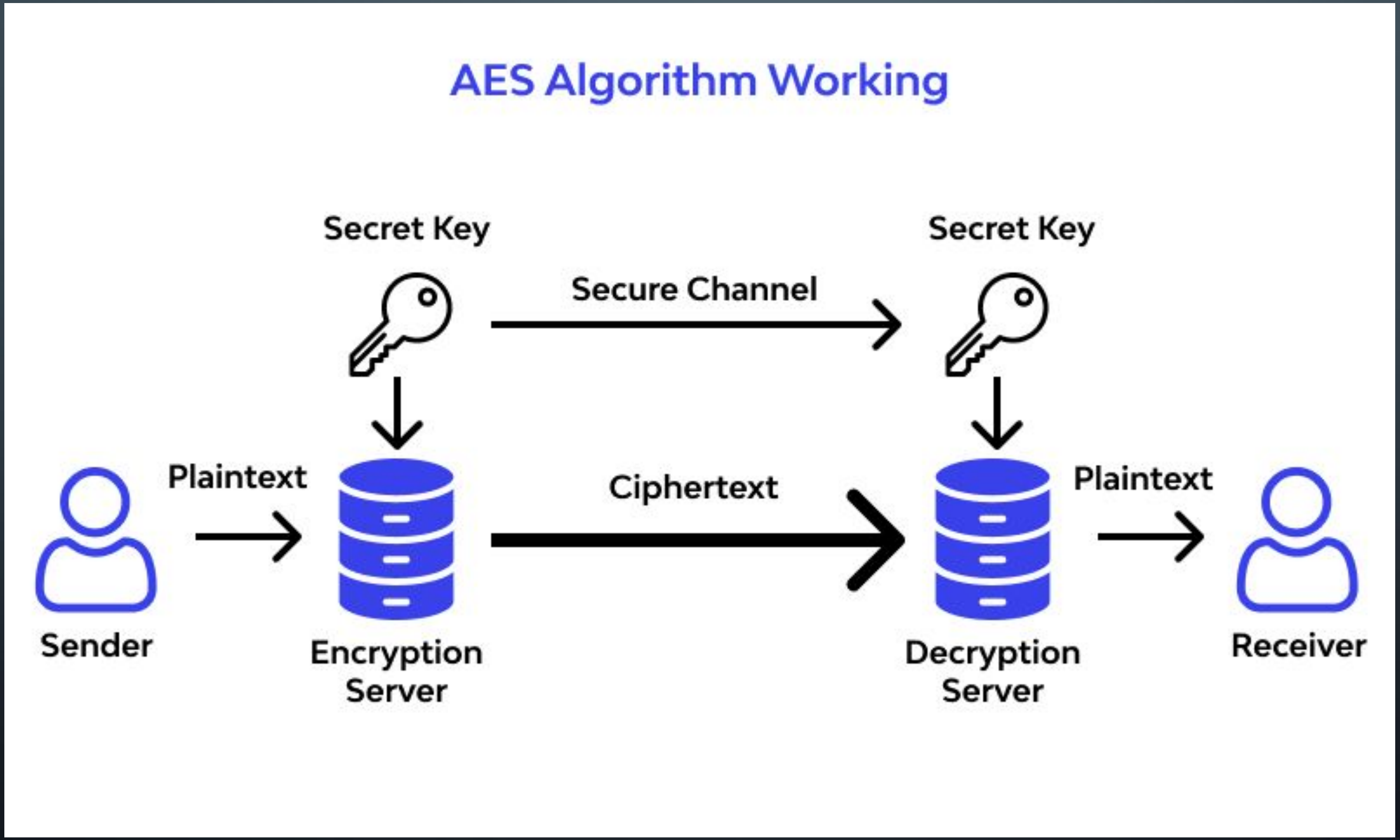
$m := \mathrm{Dec}_k(c)$

Decryption

AES

- Advanced encryption standard (AES)
  - Standardized by NIST in 2000 based on a public, worldwide competition lasting over 3 years
  - Block length = 128 bits
  - Key length = 128, 192, or 256 bits
- No real reason to use anything else

AES


AES Algorithm Working
Secret Key
Secure Channel
Secret Key
Plaintext
Ciphertext
Plaintext
Sender
Encryption Server
Decryption Server
Receiver

HASH FUNCTIONS

- (Cryptographic) hash function: maps arbitrary length inputs to a short, fixed-length digest
- Can define keyed or unkeyed hash functions – Formally, keyed hash functions are needed – In practice, hash functions are unkeyed (We will work with unkeyed hash functions, and be less formal)

COLLISION-RESISTANCE

Let H: {0,1}* → {0,1}^n be a hash function

- A collision is a pair of distinct inputs x, x’ such that H(x) = H(x’)
- H is collision-resistant if it is infeasible to find a collision in H

GENERIC HASH-FUNCTION ATTACKS

What is the best "generic" collision attack on a hash functioon

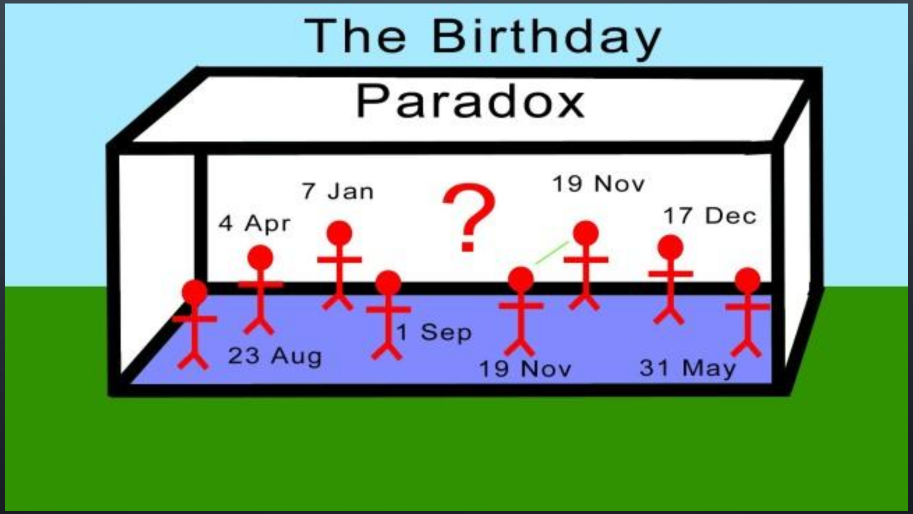
H: $\{0,1\}^* \rightarrow \{0,1\}^n$?

If we compute $H(x_1), \ldots, H(x_{2^n+1})$, we are guaranteed to find a collision – Is it possible to do better?

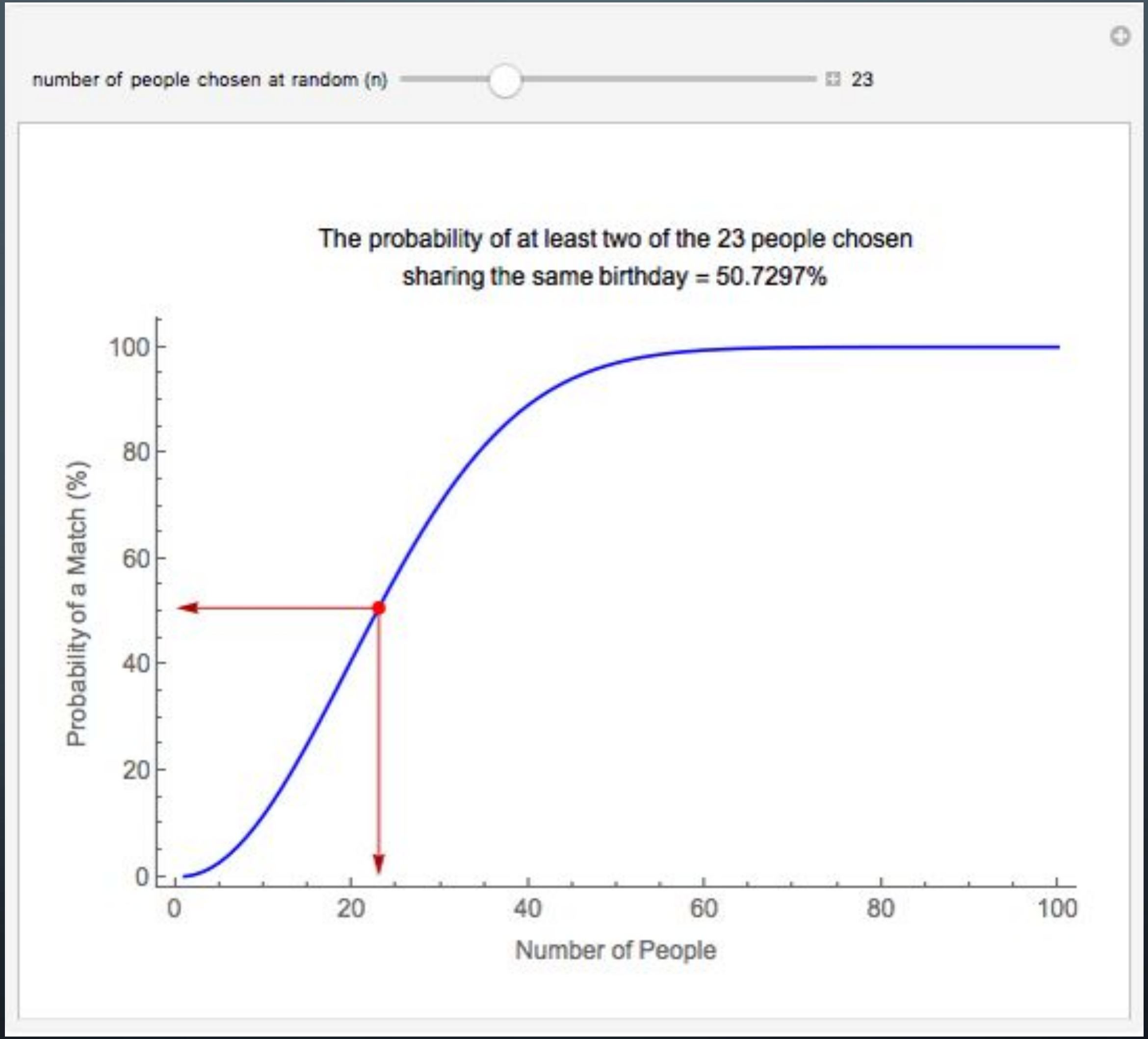
"BIRTHDAY" ATTACKS

Compute $H(x_1), \ldots, H(x_{2^{(n/2)}})$ – What is the probability of a collision?

- Related to the so-called birthday paradox – How many people are needed to have a 50% chance that some two people share a birthday?



BIRTHDAY PARADOX



number of people chosen at random (n) 23

The probability of at least two of the 23 people chosen sharing the same birthday = 50.7297%

HASH FUNCTIONS IN PRACTICE

- MD5

– Developed in 1991

– 128-bit output length

– Collisions found in 2004, should no longer be used

- SHA-1

– Introduced in 1995

– 160-bit output length

– Theoretical analyses indicate some weaknesses

– Very common; current trend to migrate to SHA-2

HASH FUNCTIONS IN PRACTICE

- SHA-2

– 256-bit or 512-bit output lengths

– No known significant weaknesses

- SHA-3/Keccak

– Result of a public competition from 2008-2012

– Very different design than SHA family

– Supports 224, 256, 384, and 512-bit outputs

HASH FUNCTIONS IN PRACTICE

BLAKE2 is a cryptographic hash function defined in RFC 7693 that comes in two flavors:

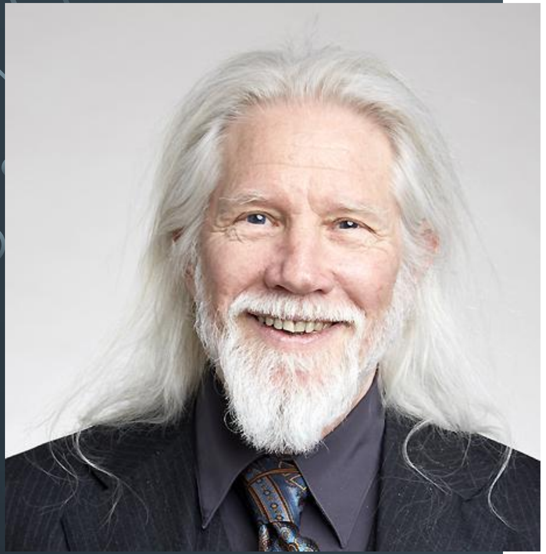
BLAKE2b, optimized for 64-bit platforms and produces digests of any size between 1 and 64 bytes,

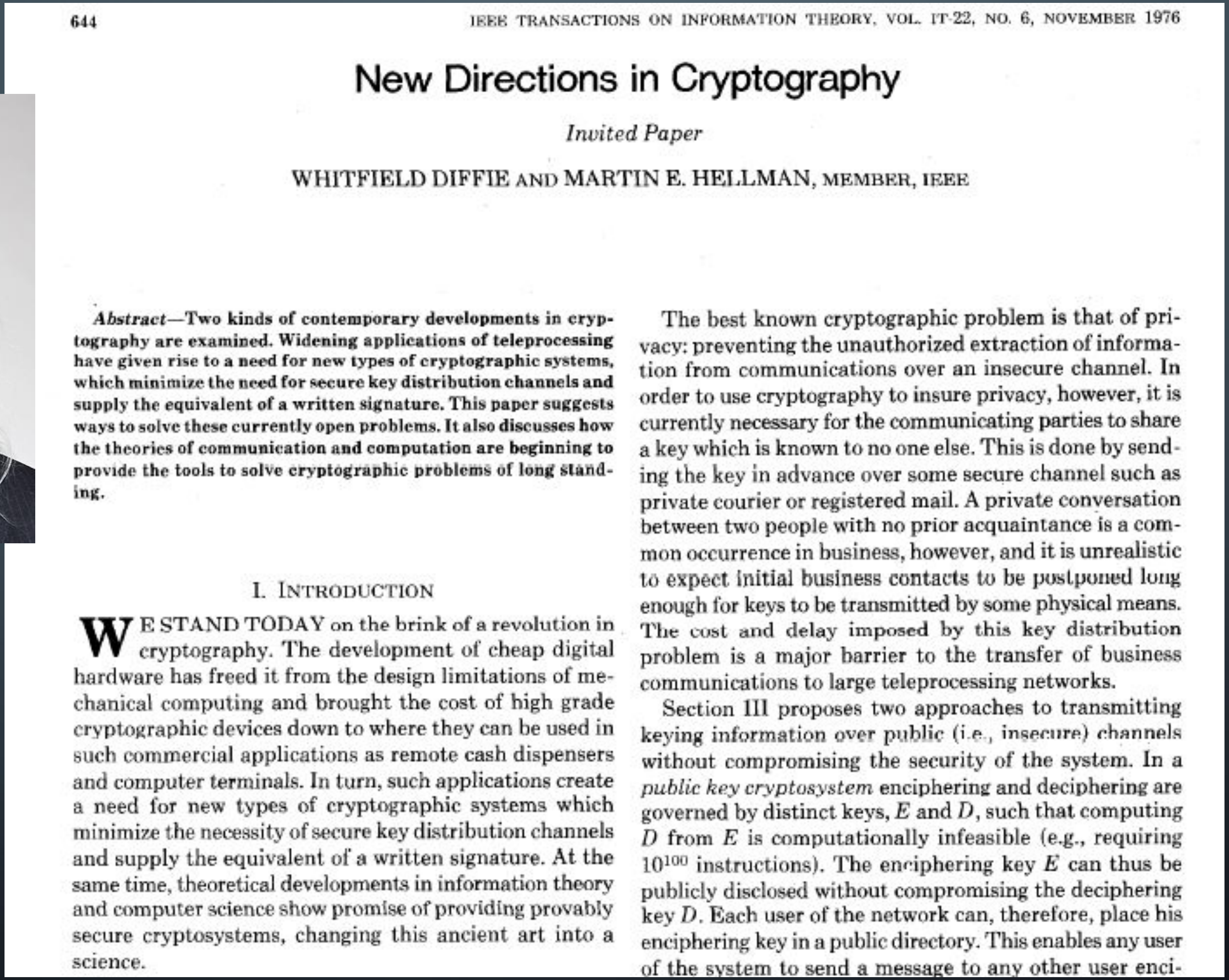
BLAKE2s, optimized for 8- to 32-bit platforms and produces digests of any size between 1 and 32 bytes.

THE KEY-DISTRIBUTION PROBLEM

- *How do users share a key in the first place?*
  - Need to share the key using a secure channel…
- This problem can be solved in some settings…
  - E.g., physical proximity, trusted courier
  - (Note: this does not make private-key cryptography useless)
- …but not others (or at least not cheaply)

“CLASSICAL” CRYPTOGRAPHY OFFERS NO SOLUTION TO THESE PROBLEMS!





New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.**

I. Introduction

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.

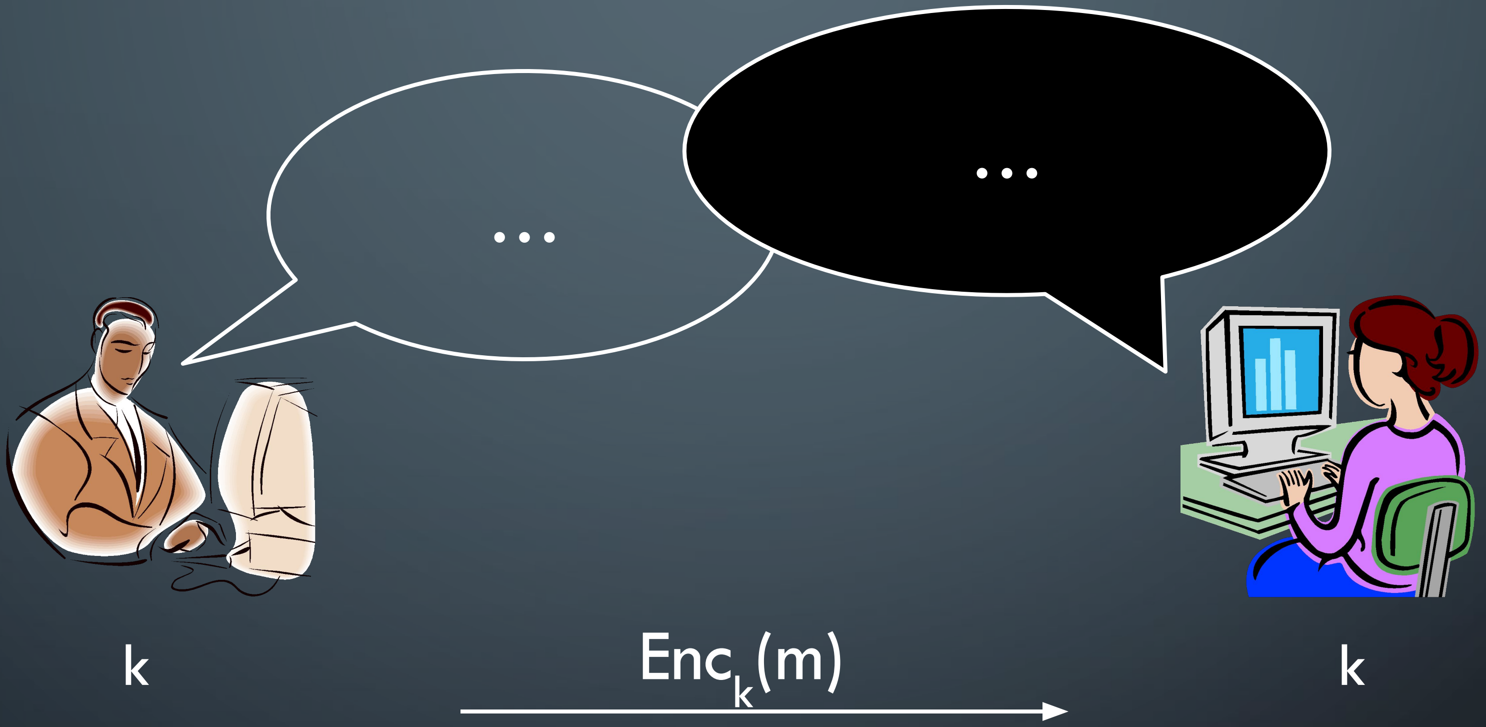
The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D, such that computing D from E is computationally infeasible (e.g., requiring 10^{100} instructions). The enciphering key E can thus be publicly disclosed without compromising the deciphering key D. Each user of the network can, therefore, place his enciphering key in a public directory. This enables any user of the system to send a message to any other user enci-

NEW DIRECTIONS…

- Key ideas:
  - Some problems exhibit *asymmetry* – easy to compute, but hard to invert (think factoring)
  - Use this asymmetry to enable two parties to agree on a shared secret key using public discussion(!)
    - *Key exchange*

KEY EXCHANGE



DIFFIE-HELLMAN KEY EXCHANGE

$$\xrightarrow{G,\ q,\ g,\ h_1}$$

$$\xleftarrow{h_2}$$

$k_1 = (h_2)^x = g^{yx}$

$(G, q, g) \leftarrow \mathcal{G}(1^n)$

$x \leftarrow \mathbb{Z}_q$

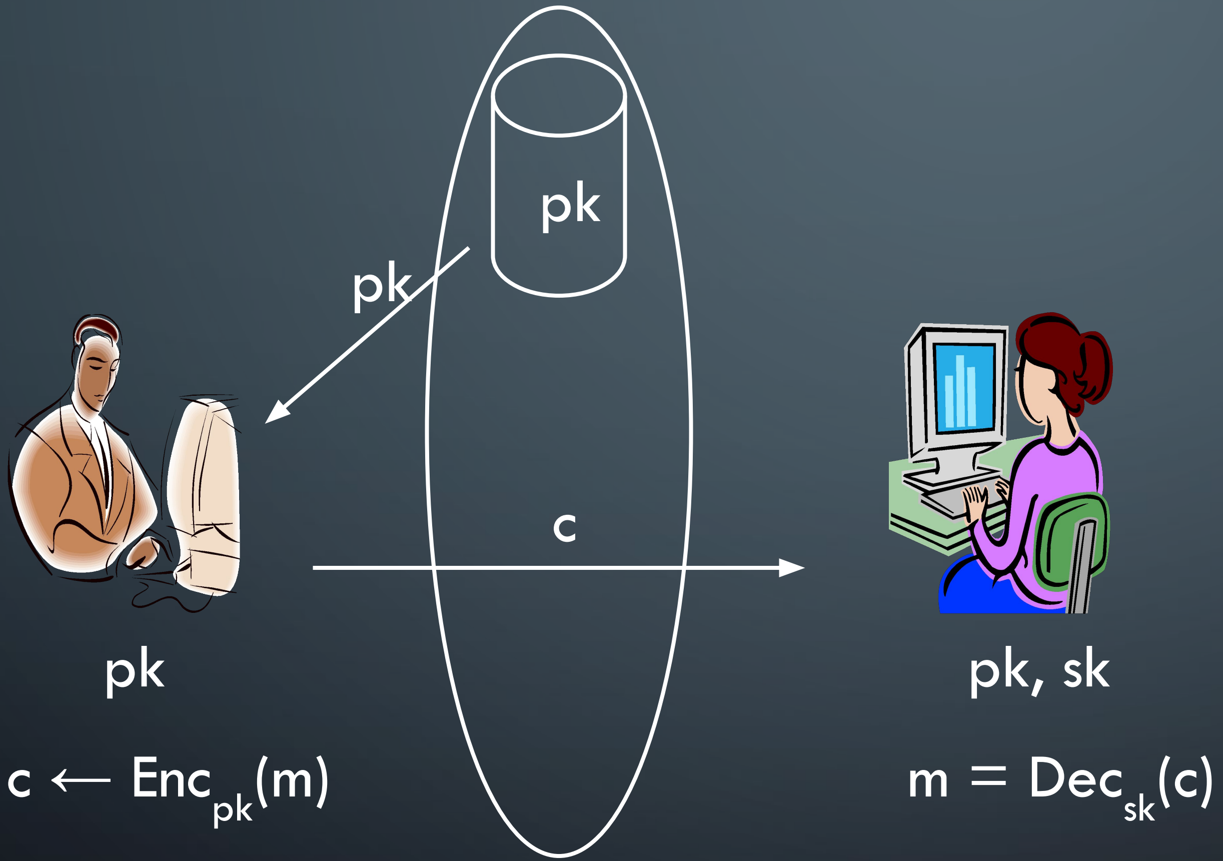
$h_1 = g^x$

$k_2 = (h_1)^y = g^{xy}$

$y \leftarrow \mathbb{Z}_q$

$h_2 = g^y$

PUBLIC-KEY ENCRYPTION


pk
pk
c
pk
pk, sk
$c \leftarrow \mathrm{Enc}_{pk}(m)$
$m = \mathrm{Dec}_{sk}(c)$

“PLAIN” RSA ENCRYPTION

- Choose random, equal-length primes p, q
- Compute modulus N=pq
- Choose e, d such that $e \cdot d = 1 \bmod \varphi(N)$

- The eth root of x modulo N is $[x^d \bmod N]$

$$(x^d)^e = x^{de} = x^{[ed \bmod \varphi(N)]} = x \bmod N$$

- *RSA assumption*: given N, e <u>only</u>, it is hard to compute the eth root of a uniform $c \in \mathbb{Z}_N^*$

“PLAIN” RSA ENCRYPTION

$$\xrightarrow{N,\ e}$$

$$\xleftarrow{c}$$

$(N, e, d) \leftarrow$ RSAGen(1^n)

pk $= (N, e)$

sk $= d$

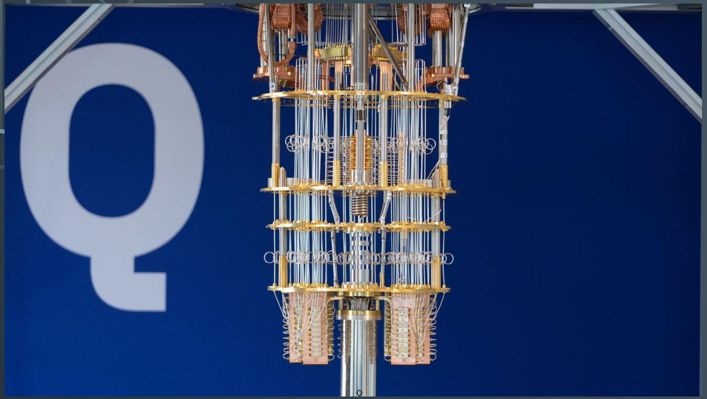
$m = [c^d \bmod N]$

$c = [m^e \bmod N]$

SECURITY?

- This scheme is *deterministic*
  - Cannot be CPA-secure!
- RSA assumptio Plain RSA should never be used! of *uniform* c
  - c is not uniform unless m is
- RSA assumption only refers to hardness of computing the eth root of c *in its entirety*
  - *Partial* information about the eth root may be leaked
  - (In fact, this is the case)

QUANTUM COMPUTERS



- GOOGLE Corporation, in conjunction with with the company D-Wave signed contract about creating quantum computers. D-Wave 2X - is the newest quantum processor, which contains physical qubits.
- Each additional qubit doubles the data search area, thus is also significantly increased the calculation speed. Quantum computers will destroy systems based on the problem of factoring integers (e.g., RSA). RSA cryptosystem is used in different products on different platforms and in different areas.

RSA system is widely used in operating systems from Microsoft, Apple, Sun, and Novell. In hardware performance RSA algorithm is used in secure phones, Ethernet, network cards, smart cards, and is also widely used in the cryptographic hardware. Along with this, the algorithm is a part of the underlying protocols protected Internet communications, including S / MIME, SSL and S / WAN, and is also used in many organizations, for example, government, banks, most corporations, public laboratories and universities.

NEWS FROM GOOGLE

- Google made a huge revelation on October 23, 2019, when it announced that it had reached something called "quantum supremacy." Via an article in the journal Nature, Google said their quantum computer, called Sycamore, solved a particularly difficult problem in 200 seconds. For comparison, Google said the world's current fastest classical computer — one called Summit owned by IBM that's as big as two basketball courts — would take 10,000 years to solve that same problem. This is what "quantum supremacy" means. It's when a quantum computer — one that runs on the laws of quantum physics as opposed to the classical computers we're familiar with (i.e. phones and laptops), which run on classical physics like Newton's laws of motion — does something that no conventional computer could do in a reasonable amount of time.

IBM'S ANSWER

- IBM responded to Google's news to say that actually, Summit could solve the quantum computers' problem in two and a half days — not 10,000 years as Google had suggested. In this episode of Recode's Reset podcast, host Arielle Duhaime-Ross and Kevin Hartnett, a senior writer for the math and physics magazine Quanta, break down exactly what quantum computing is and why Google dunking on IBM both was and wasn't a huge deal.

CHINESE RESEARCHERS ACHIEVE QUANTUM ADVANTAGE IN TWO MAINSTREAM ROUTES

- Chinese research teams have made marked progress in superconducting quantum computing and photonics quantum computing technology, making China the only country to achieve quantum computational advantage in two mainstream technical routes, while the US has only achieved a "quantum advantage" in superconducting quantum computing, analysts say.
- "Zuchongzhi 2.1," is 10 million times faster than the current fastest supercomputer and its calculation complexity is more than 1 million times higher than Google's Sycamore processor. It's the first time that China has reached quantum advantage in a superconducting quantum computing system.

RSA ALTERNATIVES

Hash-based Digital Signature Schemes: The safety of these systems depends on the security of cryptographic hash functions.

A code-based public-key encryption system: McEliece example.

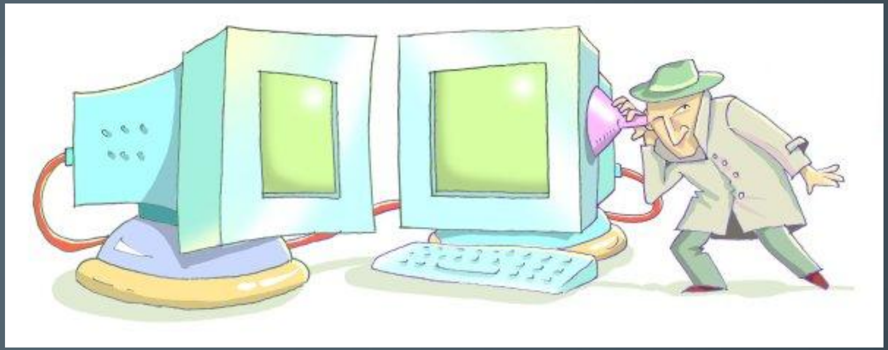
Lattice-based Cryptography: proofs are based on worst-case hardness.

Multivariate public key cryptosystem – MPKCs: have a set of(usually) quadratic polynomials over a finite field.

SUCCESSFUL ATTACKS

- To date are already found successful attacks on this crypto system.
- The Ph.D. candidate of Dublin City University (DCU) Neill Costigan with the support of Irish Research Council for Science, Engineering and Technology (IRCSET), together with professor Michael Scott, Science Foundation Ireland (SFI) member successfully were able to carry out an attack on the algorithm. To do this they needed 8,000 hours of CPU time. In the attack representatives of four other countries took part. Scientists have discovered that the initial length of the key in this algorithm is insufficient and should be increased.
- This system cannot be also used to encrypt the same message twice and to encrypt the message when is known it’s relation with the other message.

- Should be noted the importance of efficiency spectrum. To date experts have reached quite good results in the speed algorithm processing. According to the investigation results it becomes clear that the proposed post-quantum cryptosystems are relatively little effective. Implementation of the algorithms requires much more time for their processing and verification.
- Inefficient cryptography may be acceptable for the general user, but it cannot be acceptable for the internet servers that handle thousands of customers in the second. Today, Google has already has problems with the current cryptography. It is easy to imagine what will happen when implementing crypto algorithms will take more time.
- The development and improvement of modern cryptosystems will take years. Moreover, all the time are recorded successful attacks on them. When is determined



- During the implementation it is necessary to ensure not only correct work of the function and the speed of its efficiency, but also to prevent any kind of leaks. Recently have been recorded successful «cache-timing» attacks on RSA and AES system, as a result of that Intel has added the AES instructions to its processors.
- McEliece system is vulnerable to attacks, related to side channel attacks. Was shown the successful timing attack on Patterson algorithm. This attack does not detect the key, but detects an error vector that can successfully decrypt the message cipher.
- As we can see, for the creation and implementation of safe and effective post-quantum cryptosystems it is necessary to fulfill the rather big work. From the foregoing it is clear that today we are not ready to transfer cryptosystems into post-quantum era. In the near future we cannot be sure in the reliability of the systems.

RSA ALTERNATIVES – HASH BASED

- Traditional digital signature systems that are used in practice are vulnerable to quantum computers attacks. The security of these systems is based on the problem of factoring large numbers and calculating discrete logarithms. Scientists are working on the development of alternatives to RSA, which are protected from attacks by quantum computer. One of the alternatives are hash based digital signature schemes. These systems use a cryptographic hash function. The security of these digital signature systems is based on the collision resistance of the hash functions that they use.

LAMPORT–DIFFIE ONE-TIME SIGNATURE SCHEME (KEY GENERATION)

- Keys generation in this system occurs as follows: the signature key X of this system consists of 2n lines of length n, and is selected randomly.
- $X= (x_{n-1}[0], x_{n-1}[1], \ldots, x_0[0], x_0[1]) \in \{0,1\}^{n,2n}$
- Verification key Y of this system consists of 2n lines of length n.
- $Y= (y_{n-1}[0], y_{n-1}[1], \ldots, y_0[0], y_0[1]) \in \{0,1\}^{n,2n}$
- This key is calculated as follows:
- $y_i[j] = f(x_i[j])$, $0<=i<=n-1$, $j=0,1$
- f – is one-way function:
- f: $\{0,1\}^n$ □$\{0,1\}^n$;

DOCUMENT SIGNATURE

- To sign a message m of arbitrary size, we transform it into size n using the hash function:
- $h(m)=hash = (hash_{n-1}, \dots , hash_0)$
- Function h- is a cryptographic hash function:
- h: $\{0,1\}^{*}$□$\{0,1\}^{n}$
- The signature is done as follows:
- sig= $(x_{n-1}[hash_{n-1}], \dots, x_0[hash_0]) \in \{0,1\}^{n,n}$
- i-th string in this signature is equals to $x_i[0]$, if i-th bit in hashed message is equal to 0. The string is equal to $x_i[1]$, if i-th bit in sign is equal to 1.
- Signature length is $\mathbf{n^2}$.

DOCUMENT VERIFICATION

To verify the signature sig = (sig_{n-1}, …, sig_0), is calculated hash of the message hash = ($hash_{n-1}$, … , $hash_0$) and the following equality is checked:

$(f(sig_{n-1}), …, f(sig_0)) = (y_{n-1}[hash_{n-1}], …, y_0[hash_0])$

If the equation is true, then the signature is correct.

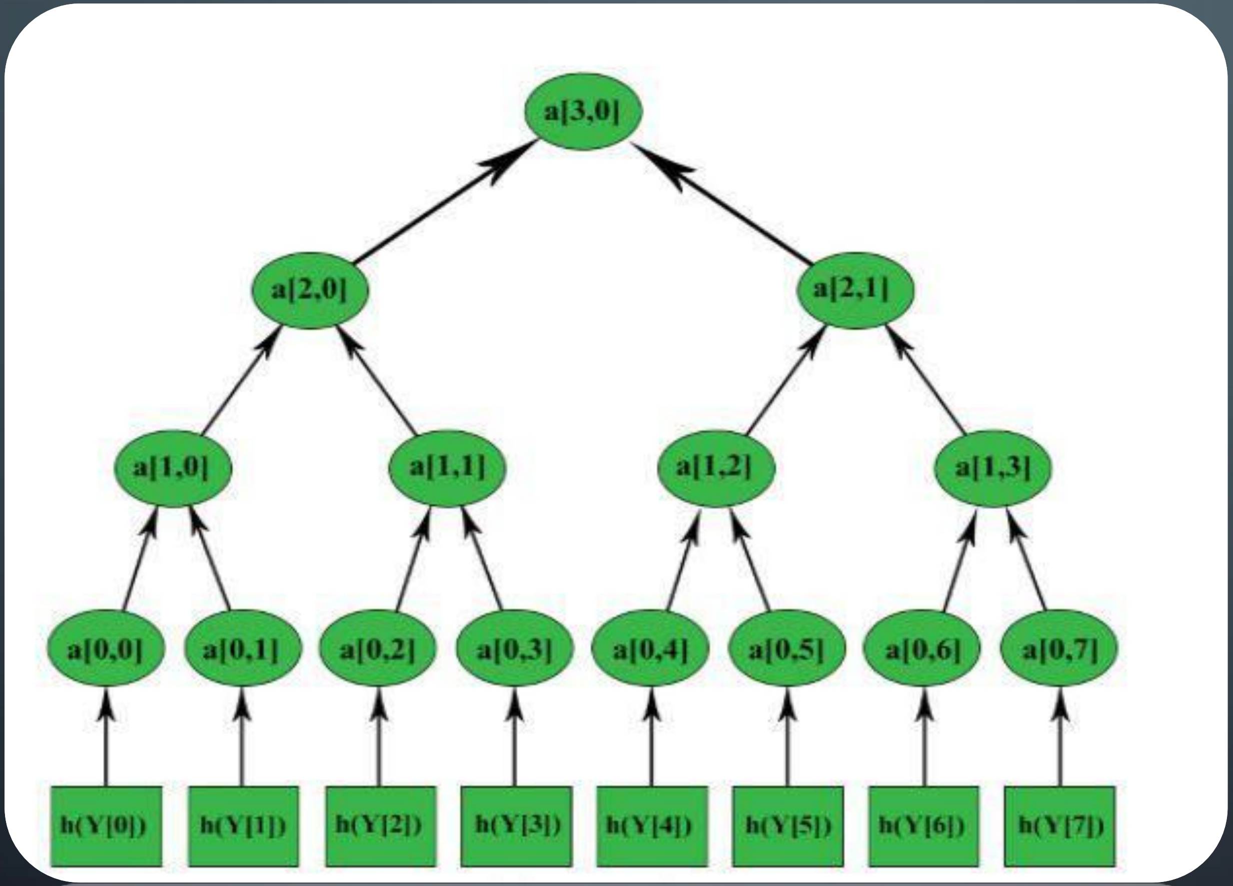
WINTERNITZ ONE TIME SIGNATURE SCHEME. KEY GENERATION

To achieve security $O(2^{80})$, the total size of public and private keys must be 160*2*160 bits = 51200 bits, that is 51200/1024=50 times larger than in the case of RSA. We must also note that the size of the signature in the given scheme is much larger than in the case of RSA. Winternitz One-time Signature Scheme was proposed to reduce the size of the signature.

MERKLE

- One-time signature schemes are very inconvenient to use, because to sign each message, you need to use a different key pair. Merkle crypto-system was proposed to solve this problem. This system uses a binary tree to replace a large number of verification keys with one public key, the root of a binary tree. This cryptosystem uses an one-time Lamport or Winternitz signature scheme and a cryptographic hash function:
- $h:\{0,1\}^*$□$\{0,1\}^n$
- **Key generation:** The length of the tree is chosen $H>=2$, with one public key it is possible to sign 2^H documents. 2^H signature and verification key pairs are generated; X_i, Y_i, $0<=i<=2^H$. X_i- is signature key, Y_i- is verification key. $h(Y_i)$ are calculated and are used as the leaves of the tree. Each tree node is a hash value of concatenation of its children.

MERKLE TREE


a[3,0]
a[2,0]
a[2,1]
a[1,0]
a[1,1]
a[1,2]
a[1,3]
a[0,0]
a[0,1]
a[0,2]
a[0,3]
a[0,4]
a[0,5]
a[0,6]
a[0,7]
h(Y[0])
h(Y[1])
h(Y[2])
h(Y[3])
h(Y[4])
h(Y[5])
h(Y[6])
h(Y[7])

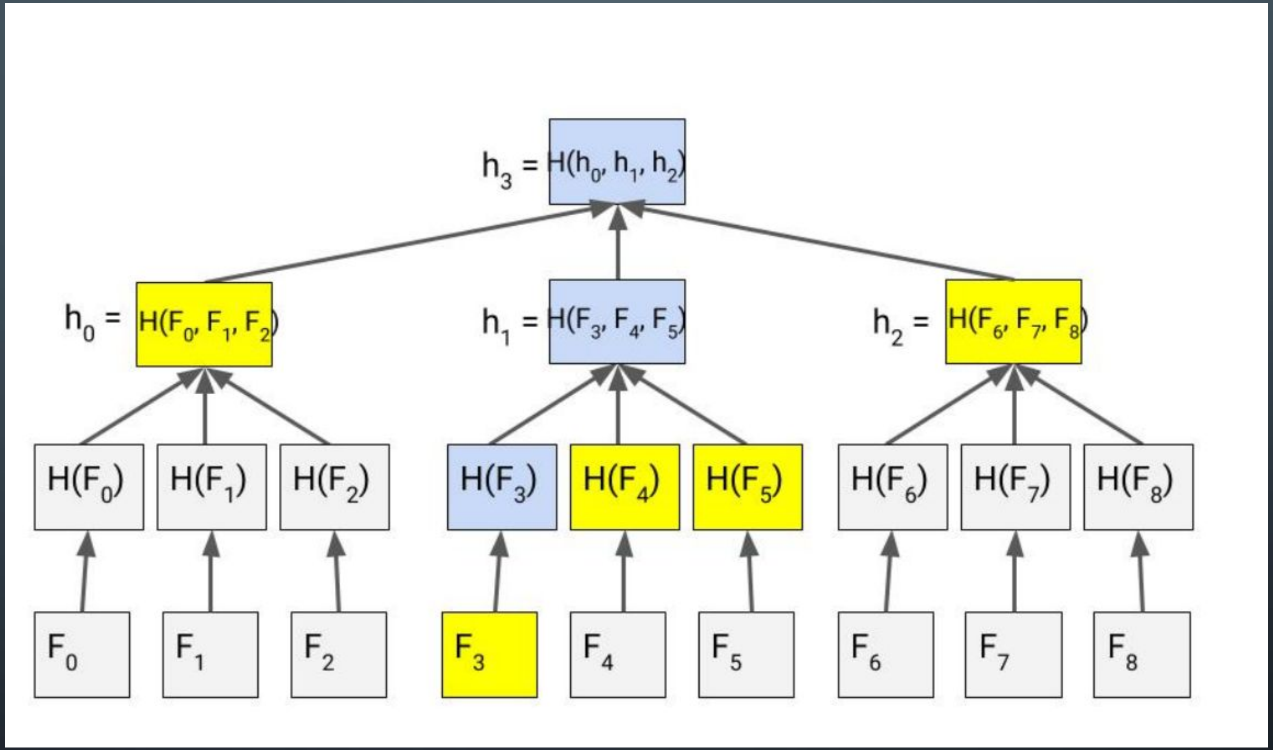
SIGNATURE GENERATION

- To sign a message m of arbitrary size we transform it into size n using the hash function
- h (m) = hash, and generate an one-time signature using any one-time key X_{any}, the document's signature will be the concatenation of: one time signature, one-time verification key Y_{any}, index any and all fraternal nodes $auth_i$ in relation to Y_{any}.
- Signature= (sig||pub||any|| Y_{any}||$auth_0$,…,$auth_{H-1}$)
- **Signature verification:**
- To verify the signature we check the one-time signature of sig using Y_{any}, if it is true, we calculate all the nodes a [i, j] using "$auth_i$", index "any" and Y_{any}. We compare the last node, the root of the tree with public key, if they are equal, then the signature is correct.

K-ARY MERKLE TREES

- One possible solution is to use a k-ary Merkle Tree. In a binary Merkle Tree, the proof consists of one node at each level, so to reduce the size of the proof, we can reduce the height of the tree by giving it a branching factor of $k > 2$.
- This approach reduces the height of the tree, but enlarges the proof size. If a branching factor is k, it reduces the height of the tree from $\log_2 n$ to $\log_k n$. $\log_2 k$ is decrease in height.
- Merkle proof actually grows larger, from $O(\log_2 n)$ to $O(k \log_k n)$.

K-ARY MERTKLE TREE

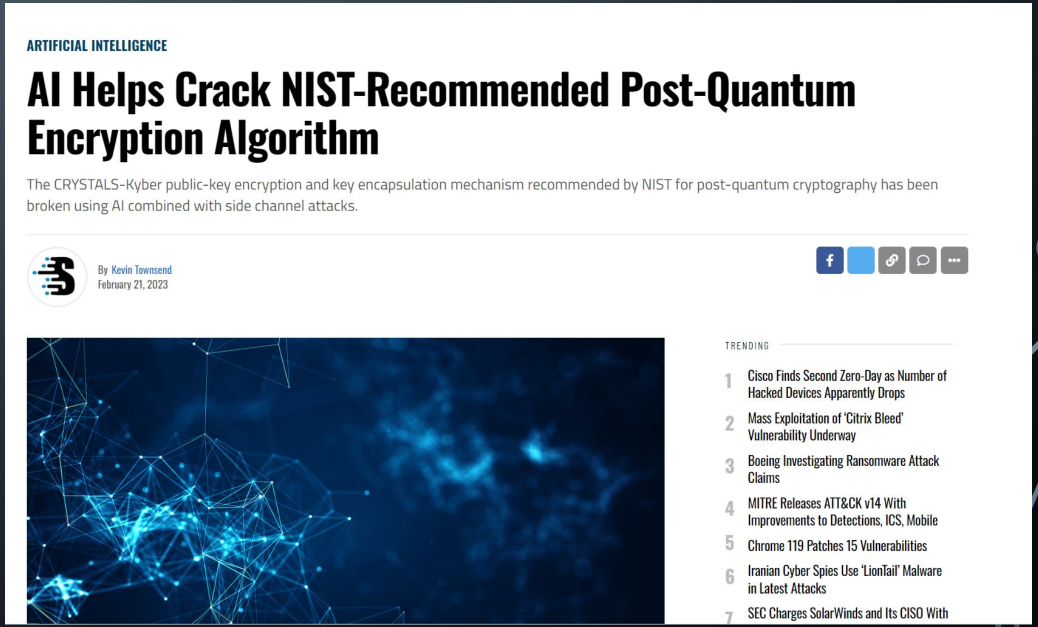


NIST

- **For general encryption**, used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation.
- **For digital signatures**, often used when we need to verify identities during a digital transaction or to sign a document remotely, NIST has selected the three algorithms CRYSTALS-Dilithium, FALCON and SPHINCS+ (read as “Sphincs plus”). Reviewers noted the high efficiency of the first two, and NIST recommends CRYSTALS-Dilithium as the primary algorithm, with FALCON for applications that need smaller signatures than Dilithium can provide. The third, SPHINCS+, is somewhat larger and slower than the other two, but it is valuable as a backup for one chief reason: It is based on a different math approach than all three of NIST’s other selections.

ATTACK- AI HELPS CRACK NIST-RECOMMENDED POST-QUANTUM ENCRYPTION ALGORITHM

- The **CRYSTALS-Kyber** public-key encryption and key encapsulation mechanism recommended by NIST in July 2022 for post-quantum cryptography has been broken. Researchers from the KTH Royal Institute of Technology, Stockholm, Sweden, used recursive training AI combined with side channel attacks.



ARTIFICIAL INTELLIGENCE

AI Helps Crack NIST-Recommended Post-Quantum Encryption Algorithm

The CRYSTALS-Kyber public-key encryption and key encapsulation mechanism recommended by NIST for post-quantum cryptography has been broken using AI combined with side channel attacks.

By Kevin Townsend
February 21, 2023

TRENDING

1 Cisco Finds Second Zero-Day as Number of Hacked Devices Apparently Drops
2 Mass Exploitation of 'Citrix Bleed' Vulnerability Underway
3 Boeing Investigating Ransomware Attack Claims
4 MITRE Releases ATT&CK v14 With Improvements to Detections, ICS, Mobile
5 Chrome 119 Patches 15 Vulnerabilities
6 Iranian Cyber Spies Use 'LionTail' Malware in Latest Attacks
7 SEC Charges SolarWinds and Its CISO With

SIDE-CHANNEL ATTACKS

- **Side-channel attacks** exploit information obtained from physically measurable, non-primary channels such as timing or power consumption of a device running the implementation.
- The researchers used a technique known as vertical side-channel leakage detection to analyze the decryption function of the CRYSTALS-Kyber algorithm. This technique involves analyzing the electrical signals produced by a computer when performing cryptographic operations. By analyzing these signals, the researchers identified weaknesses in the algorithm that could be exploited using a side-channel attack.

MASKING

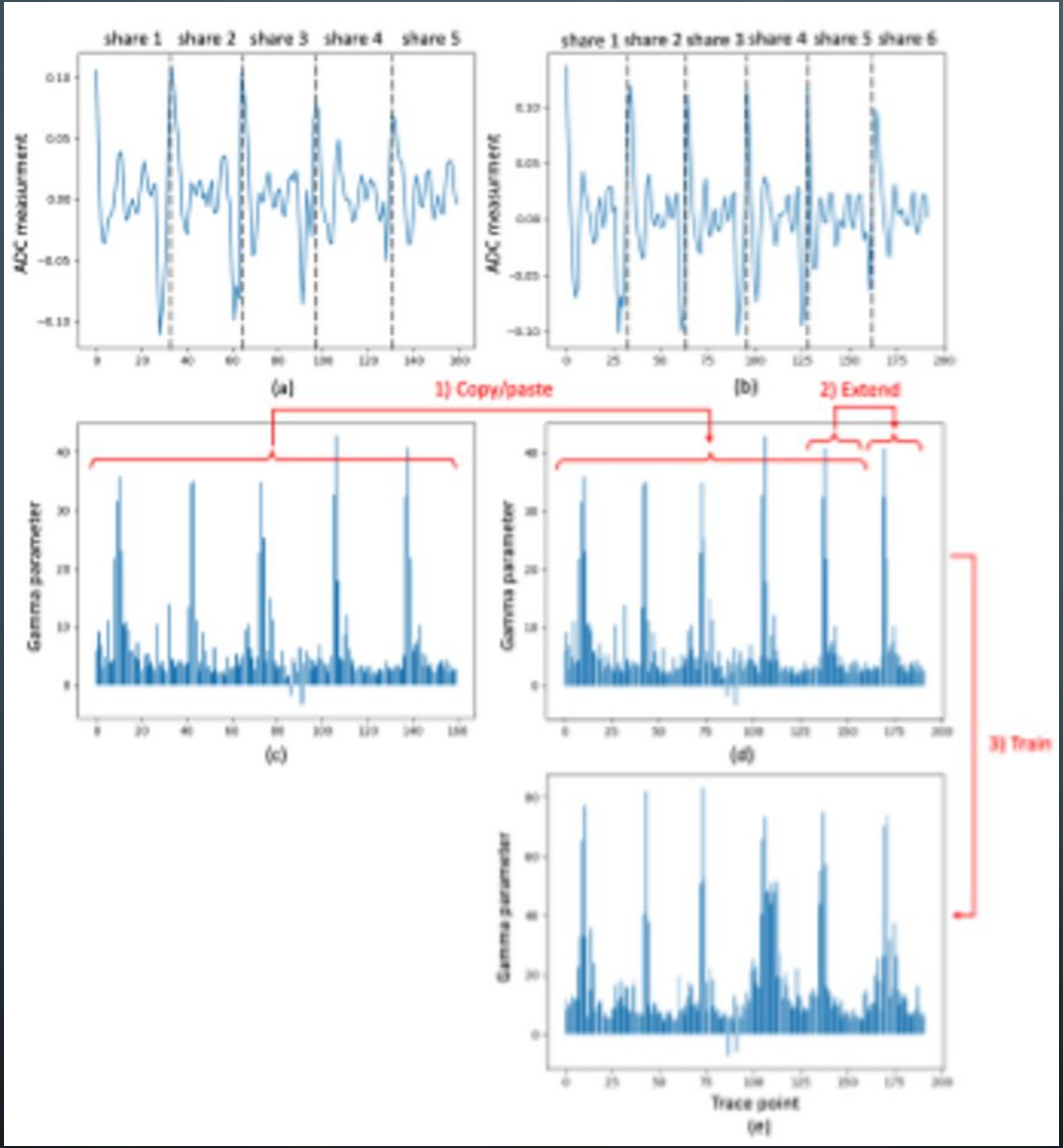
- To make CRYSTALS-Kyber resistant to side-channel attacks, a method known as masking wasused.
- Put simply, this approach randomly splits a secret into several shares, so an attacker must gather all of them to rebuild the secret. Higher-order masking is when more and more random values (i.e., masks) are used to protect a sensitive value. Specifically, an n-order masked implementation uses n+1 random values to protect each sensitive value. For example, a fifth-order masked implementation would use six random values to protect each sensitive value.

MASKING

- No higher-order implementations of CRYSTALS-Kyber are publicly available. The existing C codebase is still a finalist—not production.
- The authors had to modify the current first-order masked C implementation of CRYSTALS-Kyber to extend it to higher orders of masking, such as fifth order.
- **In other words, the researchers literally created the code version they attacked!** Yes, the researchers are trying to spot a future weakness, but this was not an attack against code that NIST released into the world. That said, there is merit to the technique, and it will need to be considered, as all potential threats must be during the torture-testing phase of a cipher's development.

CONTRIBUTIONS

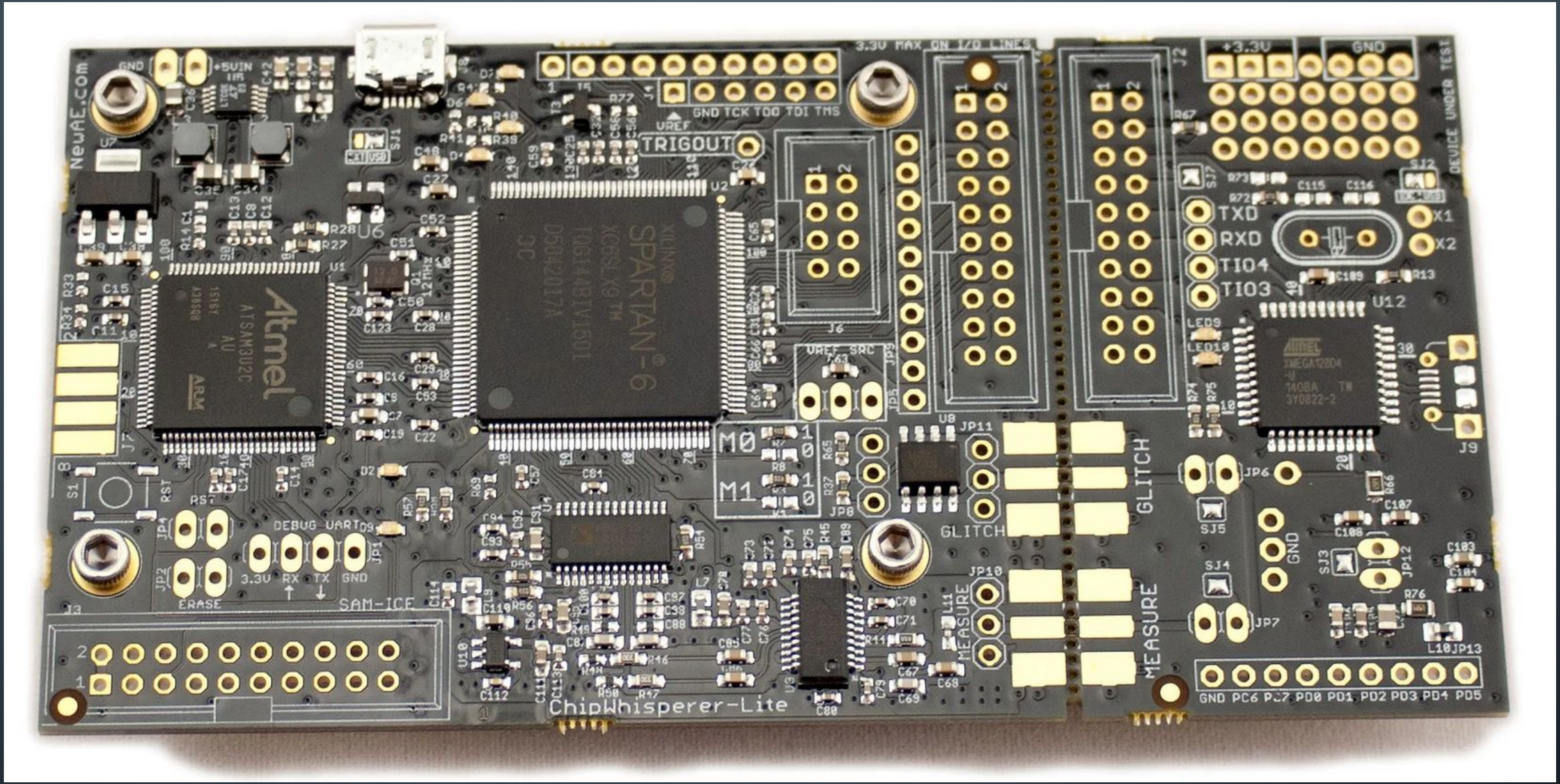
- The central idea of the work, performed with power measurement traces from an ARM Cortex-M4 MCU, involved a new neural network training technique called **recursive learning (colloquially: copy-paste).** This technique involves copying weights from an existing working model targeting less masking into a new model targeting more masking.
- Thus, a first order solution (which was presented in 2021) is used to bootstrap a second order solution and so forth. This is a particularly intriguing use of transfer learning.


share 1
share 2
share 3
share 4
share 5
share 6
ADC measurment
Gamma parameter
(a)
(b)
(c)
(d)
(e)
1) Copy/paste
2) Extend
3) Train
Trace point

CONTRIBUTIONS

- Another novel contribution is a message recovery method using **cyclic rotations**.
- In the procedure that is our attack point, the first bit of each message byte leak considerably stronger than the last one. We negacyclically rotate the message to shift its bits from "less leaky" positions to "more leaky" ones. This allow us to increase the success rate of message recovery. The messages are rotated by manipulating the corresponding ciphertexts.

EFFECTIVENESS



To test the attack, they use a Chipwhisperer-lite board, which has a Cortex M4 CPU, which they downclock to 24Mhz. Power usage is sampled at 24Mhz, with high 10-bit precision.

EFFECTIVENESS

- **To train the neural networks, 150,000 power traces are collected for decapsulation of different ciphertexts (with known shared key)** for the same KEM keypair. **This is already a somewhat unusual situation for a real-world attack**: for key agreement KEM keypairs are ephemeral; generated and used only once. Still, there are certainly legitimate use cases for long-term KEM keypairs, such as for authentication, HPKE, and in particular ECH.
- The training is a key step: different devices even from the same manufacturer can have wildly different power traces running the same code. Even if two devices are of the same model, their power traces might still differ significantly.

OUR GOAL

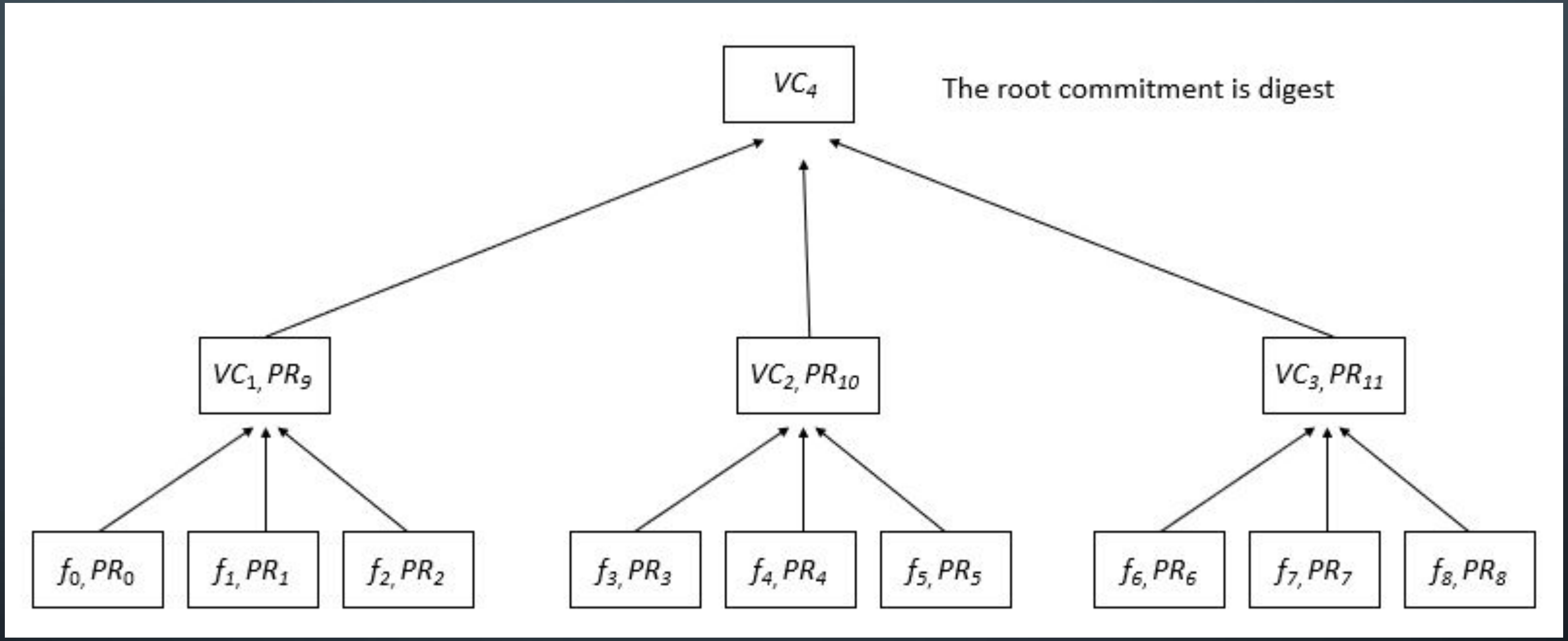
- Our goal is reduce:
- 1. the height
- 2. the proof size

VECTOR COMMITMENT TREE

- In Merkle Tree, we replace the Hash functions with the corresponding Vector Commitments.
- To compute a Verkle Tree for the messages, $m_0, m_1, \ldots, m_n$:

1. The branching factor of the tree is selected, **k**.
2. We group our messages into subsets of k and calculate a Vector Commitment, **VC**, over each of the subsets.
3. We compute each membership proofs $\mathbf{p}_i$ for every message m_i in the subset with respect to VC.
4. After we continue computing Vector Commitments up the tree over previously computed commitments until we compute the root commitment of Verkle tree.

VERKLE TREE


VC_4
The root commitment is digest
VC_1, PR_9
VC_2, PR_{10}
VC_3, PR_{11}
f_0, PR_0
f_1, PR_1
f_2, PR_2
f_3, PR_3
f_4, PR_4
f_5, PR_5
f_6, PR_6
f_7, PR_7
f_8, PR_8

COMPLEXITY

Scheme	Construction	Proof size
Merkle Tree	O(n)	$O(\log_2 n)$
k-ary Merkle Tree	O(n)	$O(k \log_k n)$
Verkle Tree	$O(n^2)$	O(1)
k-ary Verkle Tree	O(kn)	$O(\log_k n)$

ALGORITHMS

- Vector commitments can be described via the following algorithms:
- VC.KeyGen (1^k ,q) Given the security parameter k and the size q of the committed vector (with q = poly (k)), the key generation outputs some public parameters pp (which implicitly define the message space M).
- VC . Com$_{pp}$ (m$_1$,...,m$_q$) On input a sequence of q messages m$_1$,...,m$_q$ ∈ M and the public parameters pp , the committing algorithm outputs a commitment string C and an auxiliary information aux .
- VC.Open$_{pp}$ (m,i,aux) This algorithm is run by the committer to produce a proof Λ_i that m is the i -th committed message.
- VC.Ver$_{pp}$ (C,m,i,Λ_i) The verification algorithm accepts (i.e., it outputs 1) onlyif Λ_i is a valid proof that C was created to a sequence m$_1$,...,m$_q$ such that m = m$_i$.

VECTOR COMMITMENTS

- 1. Vector Commitment Based on CDH
- 2. Vector Commitment Based on RSA

PROBLEMS

- Vector commitments based on CDH and RSA can be broken by quantum computers
- Polynomial commitments based on elliptic curves can be broken by quantum computers

LATTICE-BASED VECTOR COMMITMENT

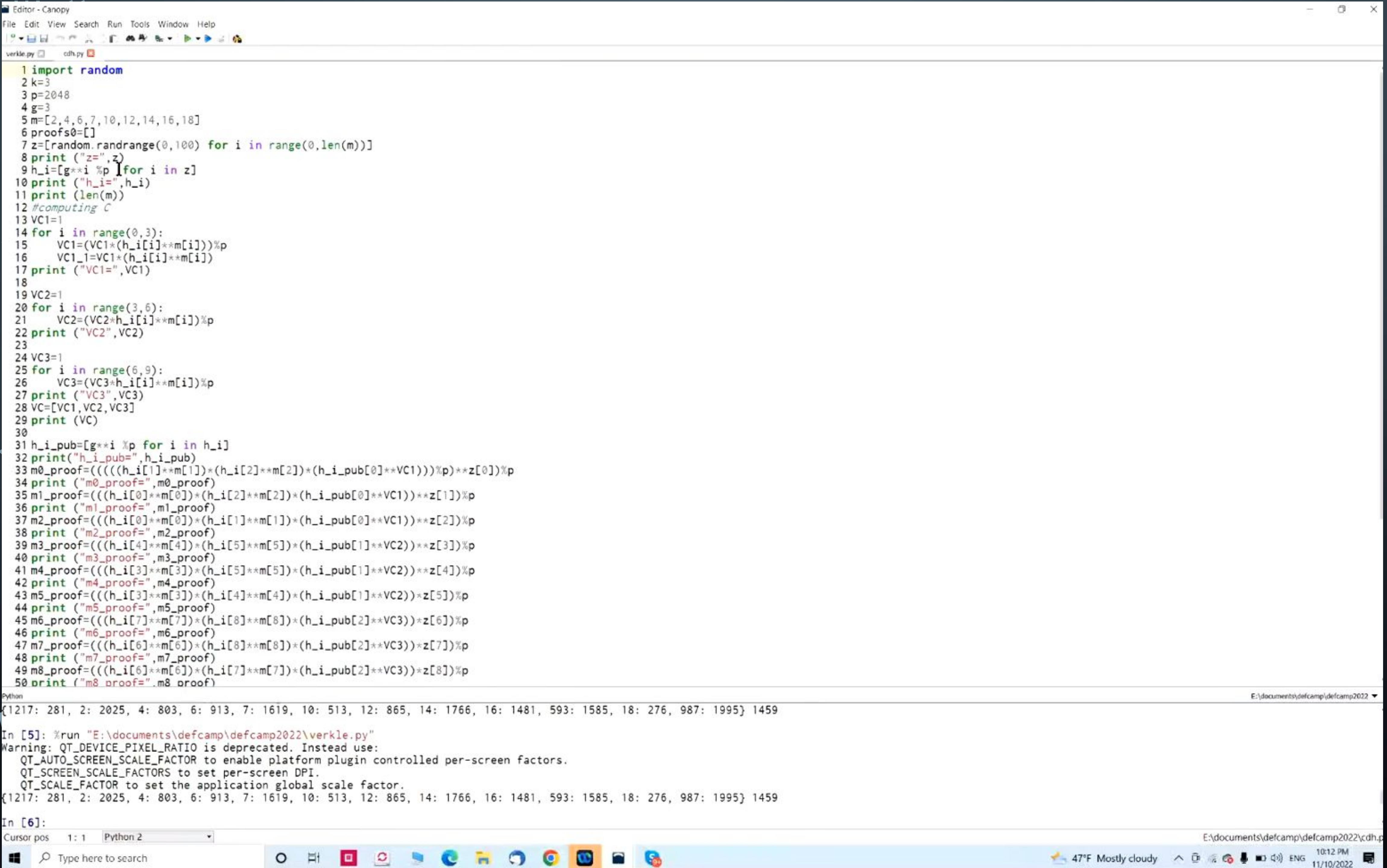
- **Setup** – This algorithm generates the committer parameters cp, and verifier parameters vp. It involves choosing a random matrix $\overline{\mathrm{A}} \leftarrow \mathbb{Z}_q^{n\times m}$, and performing the TrapGen algorithm to obtain matrices A and T. The algorithm constructs A_i matrices and a random matrix U, where each U_j is in $\mathbb{Z}_q^{n\times \ell}$. $\mathrm{R}_{i,j}$ matrices are derived using the SamplePre algorithm, ensuring that $\mathrm{H}_d - \mathrm{H}_i$ is invertible. The output of the Setup algorithm is $cp = \left(\mathrm{U}, \mathrm{R} = \left(\mathrm{R}_{i,j}\right)_{i,j\in[d]}\right)$, and $vp = (\mathrm{A}, \mathrm{U})$.
- **Commit** - Given the committer parameters cp and a message m from the message space M^d, this algorithm computes the commitment c as the sum of element-wise products of U and m. The state st is set to the message m.

LATTICE-BASED VECTOR COMMITMENT

- **Open** - This algorithm takes the committer parameters cp, the committer state st, and an index i, and computes the proof pr_i as the element-wise product of $\mathrm{R}_{i,j}$ and m_j, where $\mathrm{R}_{i,j}$ is the j - th row of the matrix R_i associated with the i - th entry of the committed message.
- **Verify** - The verifier algorithm takes the verifier parameters vp, the commitment c, the index i, the message m_i, and the proof pr_i as input. It verifies the proof by checking the conditions $\|\mathrm{p}_i\| \leq \gamma$, and $\mathrm{c} = \mathrm{A}_i\mathrm{p}_i + \mathrm{U}_i\mathrm{m}_i$. Here, γ is a security parameter. If the conditions are met, the algorithm accepts the proof, otherwise it rejects it.

NOVEL SCHEME USING VERKLE TREE

- Instead of the Merkle tree, we use the Verkle tree.
- When generating the key pair, the signer chooses $H \in \mathbb{N}, H \geq 2$. Then the key pair is generated. Using them, it will be possible to sign/verify 2^H documents. The signer will generate 2^H unique key pairs $(X_j, Y_j), 0 \leq j < 2^H$. Here, X_j is the signature key and Y_j is the verification key. Both of them are bit strings. The leaves of the Verkle tree are $g(Y_j)$, $0 \leq j < 2^H$. They are computed and used as the leaves of the tree and each node in the tree is a hash value of of its children's concatenation.
- The public key of the Verkle crypto system is the root commitment. To generate public key 2^H pairs of keys must be computed.



```python
import random
k=3
p=2048
g=3
m=[2,4,6,7,10,12,14,16,18]
proofs0=[]
z=[random.randrange(0,100) for i in range(0,len(m))]
print ("z=",z)
h_i=[g**i %p for i in z]
print ("h_i=",h_i)
print (len(m))
#computing C
VC1=1
for i in range(0,3):
    VC1=(VC1*(h_i[i]**m[i]))%p
    VC1_1=VC1*(h_i[i]**m[i])
print ("VC1=",VC1)

VC2=1
for i in range(3,6):
    VC2=(VC2*h_i[i]**m[i])%p
print ("VC2",VC2)

VC3=1
for i in range(6,9):
    VC3=(VC3*h_i[i]**m[i])%p
print ("VC3",VC3)
VC=[VC1,VC2,VC3]
print (VC)

h_i_pub=[g**i %p for i in h_i]
print("h_i_pub=",h_i_pub)
m0_proof=(((((h_i[1]**m[1])*(h_i[2]**m[2])*(h_i_pub[0]**VC1)))%p)**z[0])%p
print ("m0_proof=",m0_proof)
m1_proof=(((h_i[0]**m[0])*(h_i[2]**m[2])*(h_i_pub[0]**VC1))**z[1])%p
print ("m1_proof=",m1_proof)
m2_proof=(((h_i[0]**m[0])*(h_i[1]**m[1])*(h_i_pub[0]**VC1))**z[2])%p
print ("m2_proof=",m2_proof)
m3_proof=(((h_i[4]**m[4])*(h_i[5]**m[5])*(h_i_pub[1]**VC2))**z[3])%p
print ("m3_proof=",m3_proof)
m4_proof=(((h_i[3]**m[3])*(h_i[5]**m[5])*(h_i_pub[1]**VC2))**z[4])%p
print ("m4_proof=",m4_proof)
m5_proof=(((h_i[3]**m[3])*(h_i[4]**m[4])*(h_i_pub[1]**VC2))*z[5])%p
print ("m5_proof=",m5_proof)
m6_proof=(((h_i[7]**m[7])*(h_i[8]**m[8])*(h_i_pub[2]**VC3))*z[6])%p
print ("m6_proof=",m6_proof)
m7_proof=(((h_i[6]**m[6])*(h_i[8]**m[8])*(h_i_pub[2]**VC3))*z[7])%p
print ("m7_proof=",m7_proof)
m8_proof=(((h_i[6]**m[6])*(h_i[7]**m[7])*(h_i_pub[2]**VC3))*z[8])%p
print ("m8_proof=",m8_proof)
```

```
{1217: 281, 2: 2025, 4: 803, 6: 913, 7: 1619, 10: 513, 12: 865, 14: 1766, 16: 1481, 593: 1585, 18: 276, 987: 1995} 1459

In [5]: %run "E:\documents\defcamp\defcamp2022\verkle.py"
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
   QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
   QT_SCREEN_SCALE_FACTORS to set per-screen DPI.
   QT_SCALE_FACTOR to set the application global scale factor.
{1217: 281, 2: 2025, 4: 803, 6: 913, 7: 1619, 10: 513, 12: 865, 14: 1766, 16: 1481, 593: 1585, 18: 276, 987: 1995} 1459

In [6]:
```

QUESTIONS?

MAKSIM IAVICH

SCIENTIFIC CYBER SECURITY ASSOCIATION ; CAUCASUS UNIVERSITY

T. +(995 595) 511355; E-mail: miavich@cu.edu.ge

Scientific&practical cyber security journal – www.journal.scsa.ge

SCIENTIFIC CYBER SECURITY ASSOCIATION

DefCamp
Where Hacking & Security Collide

კავკასიის უნივერსიტეტი
CAUCASUS UNIVERSITY