Open questions

Notes by Henry Corrigan-Gibbs and Yael Kalai

MIT - 6.5610
Lecture 21 (April 28, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

The plan today is to revisit each of the topics that we covered in the course, and to look at a few of the open research problems in each area. My goals today are to (1) entertain you and (2) convince you that cryptography is one of the most exciting research areas in computer science.

One-way functions, PRGs, PRFs, etc.

The revolutionary idea of modern crypto is to base the security of cryptosystems on the conjectured hardness of some computational problem. While cryptographers had implicitly used this idea for centuries, it was not until the field of computational complexity developed—along with the notions of the complexity classes P, NP, etc.—that we could give clean formalizations of these notions.

Towards constructing encryption systems with short keys, we defined one-way functions, pseudorandom functions (PRF), pseudorandom permutations (PRP), and pseudorandom generators (PRG). There are a number of deep open questions even about these simple objects. Let me mention just a few, in decreasing order of depth.

One surprise is that it *is* possible to construct information-theoretically secure one-time MACs with short keys. This construction, due to Wegman and Carter [11], inspired the "Galois MAC" that AES-GCM uses.

Early cryptographers never (as far as I know) based the security of their cryptosystems on explicit computational assumptions. Of course, the vast majority of cryptosystems—the one-time pad being the exception—were implicitly based on hard computational problems.

In Shannon's era, we essentially classified computational problem as "computable" (e.g., factoring) or "not computable" (e.g., the Halting problem). It makes sense, then, that Shannon's notion of security for a cryptosystem required security in the face of an adversary who could compute any computable function.

Do one-way functions exist?

Most of us believe one-way functions exist, even if we have no idea how to prove it. Recall that

$$(\mathsf{P} = \mathsf{NP}) \Rightarrow \nexists\mathsf{OWF} \qquad \text{therefore} \qquad \exists\mathsf{OWF} \Rightarrow (\mathsf{P} \neq \mathsf{NP}).$$

Even $\mathsf{P} \neq \mathsf{NP}$ *is not enough.* Proving that one-way functions exist requires proving something *stronger* than $\mathsf{P} \neq \mathsf{NP}$. More specifically, $\mathsf{P} \neq \mathsf{NP}$ only implies that there *exist* hard instances of certain computational problems (e.g., 3SAT). But to construct a one-way function, we need to come up with a computational problem that is hard *on most inputs* (i.e., "hard on average") and for which we can efficiently sample an instance along with its "solution."

Russell Impagliazzo [7] has a really nice paper on these relationships.

Think of the pair $(x, f(x))$, for a one-way function f, as a solution-problem pair.

More concretely, think of any one-way function $f\colon \{0,1\}^n \to \{0,1\}^m$ that takes as input n random bits and outputs an m-bit string. The task is: given $f(x)$ recover an x' such that $f(x') = f(x)$. It could be that all functions f are easy to invert as one-way functions. More formally, for all choices of the security parameter n and for all functions f it could be that there is a p.p.t. algorithm $\mathcal{A}$ such that:

$$\Pr\left[f(x') = f(x) \colon \begin{array}{c} x \xleftarrow{R} \{0,1\}^n \\ x' \xleftarrow{R} \mathcal{A}(f(x)) \end{array}\right] > 1/\operatorname{poly}(n).$$

However, it could be that for many functions f there is no adversary $\mathcal{A}$ that inverts a *random point in the image* of f. That is, there exist functions f such that for all choices of the security parameter n there does not exist a p.p.t. algorithm $\mathcal{A}$ where

$$\Pr\left[f(x') = y \colon \begin{array}{c} y \xleftarrow{R} \{0,1\}^m \\ x' \xleftarrow{R} \mathcal{A}(y) \end{array}\right] > \frac{1}{2}.$$

This would be very sad: We still would not have one-way functions but our favorite computational problems (e.g., 3SAT) could still be hard.

Outlook. Since I do not expect to see a resolution to the P-versus-NP problem any time soon, I really have no hope of seeing a proof that one-way functions exist unconditionally. The strange thing, of course, is that it seems very easy to construct one-way functions: If you cook up any crazy function on n bits, for n large enough, there is a good chance that it will be hard to invert on random inputs. (But if you have learned anything in this course, I hope that you have learned to not try to cook up your own one-way functions.)

Scott Aaronson has a really nice paper on the P versus NP problem that I would recommend if you are interested in this question. The paper is at: `https://www.scottaaronson.com/papers/indep.pdf`.

Can we build key exchange from one-way functions?

Another one of the famous open problems in cryptography, that also seems very difficult to solve, is to construct a key-exchange protocol from any one-way function. That is, if I give you a circuit implementing an arbitrary one-way function can you use that circuit to construct a key-exchange protocol that is secure if the underlying one-way function is?

As we have seen in this course, we know how to construct key-exchange protocols from the hardness of the Computational Diffie-Hellman problem, RSA, factoring, LWE, and many other assumptions. But can we construct a key-exchange protocol whose security is based only on the hardness of inverting a one-way hash function?

As I mentioned at the start of the course, Ralph Merkle was an undergrad when he defined the notion of public key exchange and constructed this protocol.

Lest you think that this is a totally absurd idea, let me present Ralph Merkle's key-exchange protocol based on hash functions. The catch is that Merkle's protocol runs in time $O(n)$ for the honest parties and the best attack runs in time $O(n^2)$, on security parameter n.

In contrast, the best attack on standard Diffie-Hellman key exchange in elliptic-curve groups runs in exponential time, roughly $2^{n/2}$.

Setup. Alice and Bob share a public hash function $H\colon [n^2] \to \mathcal{Y}$ which we model as a random oracle, and where $\mathcal{Y}$ is some set of exponentially large size.

We use the convenient notation $[x] = \{1, \ldots, x\}$.

Merkle's Key-Exchange Protocol.

1. Alice picks n numbers $a_1, \ldots, a_n \xleftarrow{R} [n^2]$.
   She sends $H(a_1), \ldots, H(a_n)$ to Bob.
2. Bob picks n numbers $b_1, \ldots, b_n \xleftarrow{R} [n^2]$.
   He sends $H(b_1), \ldots, H(b_n)$ to Alice.
3. Alice and Bob find the least $i, j \in [n]$ such that

   $$H(a_i) = H(b_j).$$

   (If no such pair exists, restart the protocol.) They use $k = a_i = b_j$ as their shared secret.

Actually, they would hash k with an independent hash function and use the result as their shared secret.

Correctness. By the Birthday Paradox, Alice and Bob will agree on a shared secret with good probability: they are each sampling n numbers from $\{1, \ldots, n^2\}$.

Security. We will not make the argument formal, but we can indeed prove security if we model the hash function H as a random oracle. In particular, any attacker that makes $o(n^2)$ queries to the hash function H has a negligible chance of recovering Alice and Bob's shared secret.

It is not possible to build a better-than-Merkle key-exchange protocol from just a random oracle [1]. But it could be possible to build a key-exchange protocol that makes "non-black-box" use of a one-way function. That is, the construction would have to do more than just invoke the one-way function as an oracle. A non-black-box construction might, for example, somehow exploit the internals of the circuit computing the one-way function. Such a construction would have to be clever in new ways, but could potentially give us a radically different type of post-quantum-secure key exchange.

Private information retrieval (PIR)

In this class, we saw a single-server private-information-retrieval scheme, in which the client communicates with a single server and the security holds under some cryptographic assumption (e.g., LWE).

Computation cost of single-server PIR

On database size n, the PIR scheme we saw requires the server to do work $\Omega(n)$ to answer each client's query. A relatively simple argument, due to Beimel, Ishai, and Malkin, shows that $\Omega(n)$ work is necessary if the server stores the database in its original form.

In contrast, a non-private database lookup takes *constant* time on a RAM machine.

A shocking recent result [8] shows that if the server stores the database in *encoded form*, it can answer client queries in time $\mathsf{polylog}(n)$! Tragically, this scheme is wildly expensive in concrete costs [9]: querying a 256 MB database would take one month of server computation!

Are there concretely efficient single-server PIR schemes that encode the database to get sublinear server-side computation cost?

Communication cost of two-server PIR

In *two-server PIR*, the client communicates with two servers and security holds information-theoretically, provided that the adversary cannot compromise both servers.

One longstanding open question is: What is the two-server information-theoretically secure PIR scheme with the lowest communication cost? Dvir and Gopi [5] give a scheme with cost $n^{O(\sqrt{\frac{\log\log n}{\log n}})}$, which is less than n^ϵ for all $\epsilon > 0$. Is there a scheme with $\mathsf{polylog}(n)$ communication cost?

If we assume the existence of one-way functions, we can get a two-server PIR scheme with communication cost $O(\lambda \log n)$ from distributed point functions [3]. Here, λ is a security parameter, and we must assume the existence of a length-doubling PRG with seed length λ and λ-bit security.

Can we get a scheme with communication cost $O(\lambda + \log n)$ from just the assumption that one-way functions exist?

If we assume LWE, then from fully homomorphic encryption, we can get a single-server scheme with $\mathsf{poly}(\lambda) + O(\log n)$ communication cost.

Aside: Distributed point functions (DPFs). DPFs are a very neat cryptographic primitive: they give a way to compress secret-shares of a one-hot vector.

"Function secret sharing" is a generalization of distributed point functions to other types of structured vectors.

A distributed point function (DPF) over a field $\mathbb{F}$ and output length n consists of algorithms:

- $\mathsf{Gen}(1^\lambda, i \in [n], v \in \mathbb{F}) \to (s_0, s_1)$.

- $\mathsf{Eval}(s_i) \to \mathbf{x} \in \mathbb{F}^n$.

Correctness for a DPF states that the shares s_0, s_1 are compressed additive shares of the vector $v \cdot \mathbf{e}_i \in \mathbb{F}^n$, where $\mathbf{e}_i \in \mathbb{F}^n$ is the vector of all zeros with a "1" in position i. That is, for all $\lambda \in \mathbb{N}$, $i \in [n]$, and $v \in \mathbb{F}$: if we take $(s_0, s_1) \leftarrow \mathsf{Gen}(1^\lambda, i, v)$, then

$$\mathsf{Eval}(s_0) + \mathsf{Eval}(s_1) = v \cdot \mathbf{e}_i \quad \in \mathbb{F}^n.$$

Security for a DPF states that each share individually hides both i and v. That is, there exists a p.p.t. simulator Sim such that for all $\beta \in \{0,1\}$, $i \in [n]$, and $v \in \mathbb{F}$:

$$\{s_\beta : (s_0, s_1) \leftarrow \mathsf{Gen}(1^\lambda, i, v)\}_\lambda \overset{c}{\approx} \{\mathsf{Sim}(1^\lambda, \beta)\}_\lambda.$$

Given a DPF, the construction of a two-server PIR scheme almost immediate: the client generates compressed secret shares of a unit vector (non-zero in the location of the database that the client wants to read) and sends the shares to the servers. The servers expand the shares, take the inner product of these shares with the database, and return the answer to the client. Summing the servers' answers gives the client its database bit of interest.

Fully homomorphic encryption (FHE)

FHE without lattices or pairings. We have two ways to construct fully homomorphic encryption today: One way is via lattice-type assumptions—LWE and friends. The other way is via indistinguishability obfuscation, which relies on coding assumptions and pairing-based cryptography. (We have not talked about pairings in this class.)

Is there a third way? Can you construct FHE from the assumption that factoring or discrete log is hard?

FHE without bootstrapping. Moreover, today's lattice-based FHE schemes all use the "bootstrapping" approach that Alexandra described in her guest lecture: The scheme performs some computation on ciphertexts until the noise level in the ciphertexts becomes too large. Then, the scheme refreshes the ciphertexts by homomorphically evaluting its own decryption circuit on the ciphertexts—this is bootstrapping.

Bootstrapping is annoying for two reasons: First, it is computationally expensive, since the decryption circuit can be large. Second, it requires an additional "circular security" assumption: we have to assume that the encryption scheme is secure even if we publish the encryption of the secret key under itself.

Can we construct FHE in a totally different way that does not require bootstrapping?

Identification protocols and signature schemes

Is there a human-computable many-time-secure authentication protocol?

Say that you get stuck in the dessert in some faraway country hundreds of miles away from the nearest ATM. (This actually happened to me one time.) You call your parents and ask them to wire you $100 via Western Union—which to my surprise works even in places without ATMs. But since your parents are cryptographers, they they want to authenticate you first.

In particular, you share a secret key k with your parents. They will read you a challenge message over the phone and you have to respond, using only a paper and pencil. We want many-time security against eavesdropping attacks: even if an attacker listens to many interactions between you and your parents, the attacker shouldn't be able to impersonate you.

If you had a PRF F, it would be easy to construct such an authentication protocol: your parents send a nonce r, you reply with $a \leftarrow F(k,r)$, and your parents check that $a = F(k,r)$. So an equivalent question is whether it is possible to construct a PRF (or even a weak variant of a PRF) that a human can compute.

The question, which I suspect Manuel Blum first posed, is whether it is possible to construct an authentication scheme that a human can implement with paper and pencil. What I like about this question is that it first requires you to understand what humans can and cannot compute. Just as we cannot talk about one-way functions without having good models of machine computation, we cannot hope to talk about human-computable one-way functions without having a good model of human computation.

Nick Hopper and Manuel Blum [6] came up with a very simple protocol that is plausibly human computable, and is based on the (very solid) learning-parities-with-noise assumption (LPN).

The LPN assumption is essentially a "scaled down" version of the LWE assumption with modulus $q = 2$. The LPN assumption is equivalent to stating that it is computationally infeasible to decode random linear codes over $\mathbb{F}_2$. This is a problem that has been studied since at least 1968 [2].

Hopper-Blum authentication protocol. The two parties' shared secret is a vector $k \in \mathbb{Z}_2^n$. Think of $n \approx 4\,096$ or so.

- **Challenge.** The challenge is a random vector $r \xleftarrow{R} \mathbb{Z}_2^n$.
- **Response.** The response is the value

$$a \leftarrow \langle k, r \rangle + \epsilon \qquad \in \mathbb{Z}_2,$$

  where $\langle k, r \rangle = \sum_{i=1}^{n} k_i \cdot r_i \in \mathbb{Z}_2$, and $\epsilon \in \{0,1\}$ is a biased coin that is 1 with probability $1/10$.
- **Accept/reject.** The verifier accepts if $a = \langle k, r \rangle \in \mathbb{Z}_2$.

If the verifier runs the protocol 1 000 times and accepts if and only if the prover (person authenticating) gives the correct answer at least

80% of the time, then the verifier can be pretty sure that they are talking to the right person.

Is this protocol really human computable? What would a better protocol look like? And what does it mean for a function to be "human computable" or not?

Short signatures

A very simple open problem is: Are there digital signature schemes (under plausible assumptions) in which the signatures are λ bits long and the best forgery attack runs in time very close to 2^λ? We have signatures from pairings/bilinear maps that have bitlength roughly 2λ that achieve λ-bit security, but getting even shorter signatures would be very useful. ECDSA signatures are closer to 3λ bits long, or roughly 384 bits when we are aiming for 128-bit security.

Cryptanalysis

Classical: Better algorithms for factoring Jim and Ron's talks touched on one of the simplest open questions cryptography: How hard is factoring? Is there a polynomial-time algorithm for it? The best algorithms today for factoring an n-bit number run in time roughly $2^{\widetilde{O}(\sqrt[3]{n})}$, which is much much better than 2^n, but is much worse than n^{100}, since the exponent grows without bound. It seems unlikely that $2^{\widetilde{O}(\sqrt[3]{n})}$ is the correct running time for the best factoring algorithm in any just world, but the tricks we know cannot do better than that.

The $\widetilde{O}(\cdot)$ notation hides log factors in its argument. That is $\widetilde{O}(f(x))$ means $O(f(x) \cdot \operatorname{polylog} f(x))$.

To get a sense of where this running time comes from: If you take a random n-bit number, the probability that all of its prime factors are less than $B = 2^{\sqrt{n}}$ is (roughly) $1/B$. We call these "B-smooth" numbers. The index-calculus-type factoring algorithms require finding (roughly) B such smooth numbers.

The most powerful factoring algorithms need to find $B = 2^{\widetilde{O}(\sqrt[3]{n})}$ numbers whose prime factors are all $\leq B$. The expected running time is then roughly $B^2 = 2^{2\cdot\widetilde{O}(\sqrt[3]{n})}$, which is as claimed.

Quantum: Faster factoring algorithms To understand this open problem, we need a tiny bit of background on quantum computation. We typically model the state of a quantum computer as a "register" that holds n qubits. (Don't worry about what a qubit is.) To run a quantum computation, we apply two- or three-input quantum "gates" (gates as in a circuit) to subsets of the qubits in the registers. The gates modify the state of the register contents. At the end of the computation, the output is in some subset of the qubits in the register.

There are two especially important complexity measures then for a quantum computation:

1. **Circuit size.** How many quantum gates must you apply to the register before getting the output of the computation?

2. **Register size.** How large of a register, in qubits, do you need to run the computation?

Large quantum computers seem extremely hard to build: speaking optimistically, the best computers today can (maaaaaybe) run certain computations with hundreds of gates on tens of qubits. So minimizing the size of a quantum computer required to factor is a neat and well-motivated open question.

Here is what we know about quantum factoring, from the timeline of Ragavan and Vaikuntanathan [10]:

- Shor's algorithm uses a quantum circuit of size $O(n^2 \log n)$ and requires a register of $O(n \log n)$ qubits.
- Regev recently gave a circuit of size $O(n^{3/2} \log n)$ using $O(n^{3/2})$ qubits.
- The most recent work [10] gives a circuit of size $O(n^{3/2} \log n)$ using $O(n \log n)$ qubits.

Is it possible to construct a quantum factoring circuit of near-linear size using a near-linear number of qubits?

Quantum: Better algorithms for lattice problems A paper last year [4] proposed a quantum polynomial-time algorithm for the LWE problem, which we have covered in detail in this course. The paper—and how long it took to refute it—underscores how little we know about quantum cryptanalysis. My sense is that precious few people (not counting people in intelligence agencies) spend any time considering the quantum hardness of these lattice problems.

Chen's LWE algorithm turned out to be flawed, but there is still a very real possibility that someone will come up with a fast quantum algorithm for the problem. The best known algorithms run in exponential time.

Is there a subexponential-time quantum algorithm for LWE?

Proof systems

Succinct proof systems have been gaining popularity over the last two decades especially with increasing large scale computations. There are many important problems in this setting. First, we know how to generate succinct non-interactive arguments (SNARGs) for the correctness of a computation.

In class we saw the GKR succinct interactive protocol for bounded depth computations, and as we mentioned we can apply the Fiat-

Shamir protocol to it, and guarantee soundness under LWE (if we use a suitable Fiat-Shamir Hash functions). There are other ways of constructing such SNARGs for computations of unbounded depth.

We also talked about SNARGs for NP, which is a way of taking any mathematical proof and shrinking it using cryptography. We have seen a construction that are sound in the Random Oracle Model. However, we still do not know of a construction from standard assumptions. This is a major open problem in cryptography.

Another philosophical question that I am fascinated by is the question of the efficiency of interactive proofs. We know that IP=PSPACE but in this IP the prover runs in time $\text{poly}(2^S)$ where S is the space of the underlying computation. Is this inherent? Can we improve the prover's overhead to be polynomial (as in GKR), and obtain a doubly-efficient version of the IP=PSPACE theorem?

Quantum Cryptography

There was a class this term (6.S895), devoted entirely to quantum cryptography.

We didn't talk much about quantum cryptography in this class. We did talk about *post-quantum* cryptography where all the honest parties are honest but where we want security even if the adversary has quantum power. Quantum cryptography is where the honest parties have quantum power too. There are countless important open questions in this regime. I would partition them into two buckets:

1. **Quantum-mania:** All parties have quantum power and they exchange quantum bits.
2. **Mini-quant:** An few parties have quantum power (these parties can be thought of as IBM, Microsoft, etc.), but most computers are classical.

For the land of quantum-mania, one of the most fundamental question is what are the minimal assumptions needed to do cryptography in this world? For example, we know that key-exchange can be done information theoretically. In the classical model of computation, it is known that the existence of a one-way function is the minimal assumption needed for essentially any cryptographic application that cannot be realized information theoretically. In the quantum setting, however, one-way functions appear not to be essential, and the question of whether such a minimal primitive exists remains open. This is an active area of research.

For the land of mini-quant, there are many fundamental questions. For one, do we have SNARGs for quantum computations? Or even simpler: Can a quantum computer prove to a classical computer that a quantum computation was done correctly (non-succinctly)?

Urmila was the first to give a positive answer to this question, by constructing an interactive argument for all quantum computations under the LWE assumption. Recently, her protocol was converted to a succinct one. We mention that without succinctness it is not clear that we need to rely on any assumption. This gives rise to the following fundamental question: Is there an interactive proof for BQP or QMA where the verifier is classical and the prover runs in quantum polynomial time?

What next?

- **Attend seminars.** We have a weekly Cryptography and Information Security seminar, a weekly security seminar, and the quarterly Charles River Crypto Day. All are free to attend and open to the public. Most of these events appear on the CSAIL calendar.
- **Take classes.** Related classes at MIT are 6.858 (Spring) and 6.875 (Fall). Yael is teaching a class on proof systems in the fall as well—course number TBD. There are many classes at Harvard relating to privacy, law, cryptography, etc. You can cross-register for these for free!
- **Get involved in research.** For undergraduates: UROPs are a great way to get started, if you haven't tried them already. If you have done a great project in this class, you can share it with your prospective research advisor. The Cryptography ePrint Archive is a place to get the latest research papers in cryptography (for free!).

Closing notes

Feel free to email me or Yael any time if you have questions about cryptography or anything else. We do not always have capacity to take on UROP/MEng projects, but we're always happy to brainstorm about research ideas and chat about whatever else is on your mind.

Have a great summer!

References

[1] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal–an $o(n^2)$-query attack on any key exchange from a random oracle. In *CRYPTO*, 2009.

[2] Elwyn R Berlekamp. *Algebraic Coding Theory*. 1968.

[3] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *CCS*, 2016.

[4] Yilei Chen. Quantum algorithms for lattice problems. Cryptology ePrint Archive, Paper 2024/555, 2024. `https://eprint.iacr.org/2024/555`.

[5] Zeev Dvir and Sivakanth Gopi. 2-server PIR with subpolynomial communication. *Journal of the ACM (JACM)*, 63(4):1–15, 2016.

[6] Nicholas J Hopper and Manuel Blum. Secure human identification protocols. In *ASIACRYPT*, 2001.

[7] Russell Impagliazzo. A personal view of average-case complexity. In *IEEE Conference on Structure in Complexity Theory*, pages 134–147. IEEE, 1995.

[8] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE. In *STOC*, 2023.

[9] Hiroki Okada, Rachel Player, Simon Pohmann, and Christian Weinert. Towards practical doubly-efficient private information retrieval. In *International Conference on Financial Cryptography and Data Security*, 2024.

[10] Seyoon Ragavan and Vinod Vaikuntanathan. Space-efficient and noise-robust quantum factoring. 2004.

[11] Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.