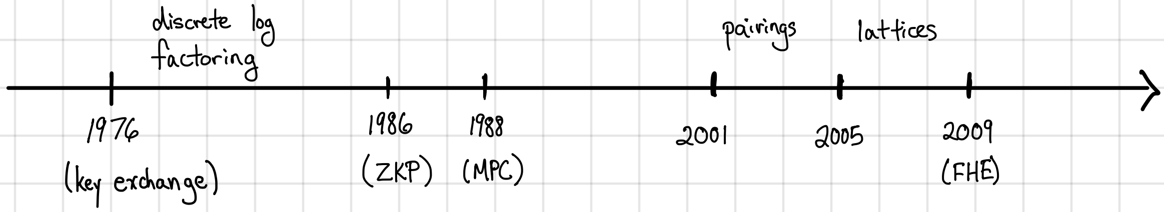
# CS 6501 Week 15: Summary and Open Problems

What this course was all about:




Early days of cryptography: How do Alice and Bob <u>communicate securely</u> on a public, untrusted network?
- ↳ Led to notions like secure key exchange, semantic security, digital signatures
- ↳ Enabled the development of the Internet as we know it today

Over time, cryptography evolved beyond protecting communication to <u>protecting computation</u>: How do Alice and Bob compute a function of their <u>secret</u> inputs?
- ↳ Began with multiparty computation (MPC), but subsequently extended to encryption schemes (e.g., FHE, ABE, functional encryption, obfuscation)

<u>Functional encryption</u>: a general umbrella for encryption
- Secret keys are associated with functions $f$
- Decryption yields a <u>function</u> of the message

$$\left.\begin{array}{l} ct_x \leftarrow \text{Encrypt}(mpk, x) \\ sk_f \leftarrow \text{KeyGen}(msk, f) \end{array}\right\} \Rightarrow \text{Decrypt}(sk_f, ct_x) = f(x)$$

Public key encryption is FE for the identity function
Attribute-based encryption is FE for the following class of functions

$$g(x, m) = \begin{cases} (x, m) & \text{if } f(x) = 1 \\ (x, \perp) & \text{if } f(x) = 0 \end{cases}$$

($x$: attribute, $m$: message, $f$: predicate)

[captures fact that attributes in ABE scheme are public]

Nice general framework for describing encryption schemes: very powerful, but difficult to construct
- ↳ But not so difficult if we only require <u>single-key</u> security [here, PKE even suffices!]

<u>Key idea</u>: Will rely on garbled circuits and use PKE to "non-interactively implement OT"
- Let $U$ be the universal circuit (for evaluating circuits of some bounded size): $U(C, x) = C(x)$
- To encrypt a message $x$, the encrypter will prepare a garbled circuit for $U$ and give out the labels for $x$
  - ↳ <u>Challenge</u>: We need a non-interactive way for the decrypter to obtain the labels for the circuit $C$ (part of the secret key)
- <u>Key idea</u>:
  - Public parameters of FE scheme will consist of $2\ell$ public keys (where $\ell$ is the description size of $C$)
  - Encrypter will encrypt wire labels for bits of $C$ under the corresponding public keys
  - Secret key for circuit $C$ will consist of decryption keys corresponding to bits of $C$
- <u>Observe</u>: If decrypter has just one decryption key, it only gets one set of labels for the garbled circuit, so by security of the garbling scheme, ciphertext can be simulated just given $C(x)$ [provided that PKE scheme is semantically secure] → secure single-key FE

Simple construction, but very powerful!

Many ways to improve:

1. Collusion-resistant FE: here, decrypter who has many keys completely breaks security
2. Compact FE: ciphertexts in this scheme scale with the size of the function
3. Multi-input FE: given two ciphertexts $ct_x$ and $ct_y$, evaluate a bivariate function on underlying messages

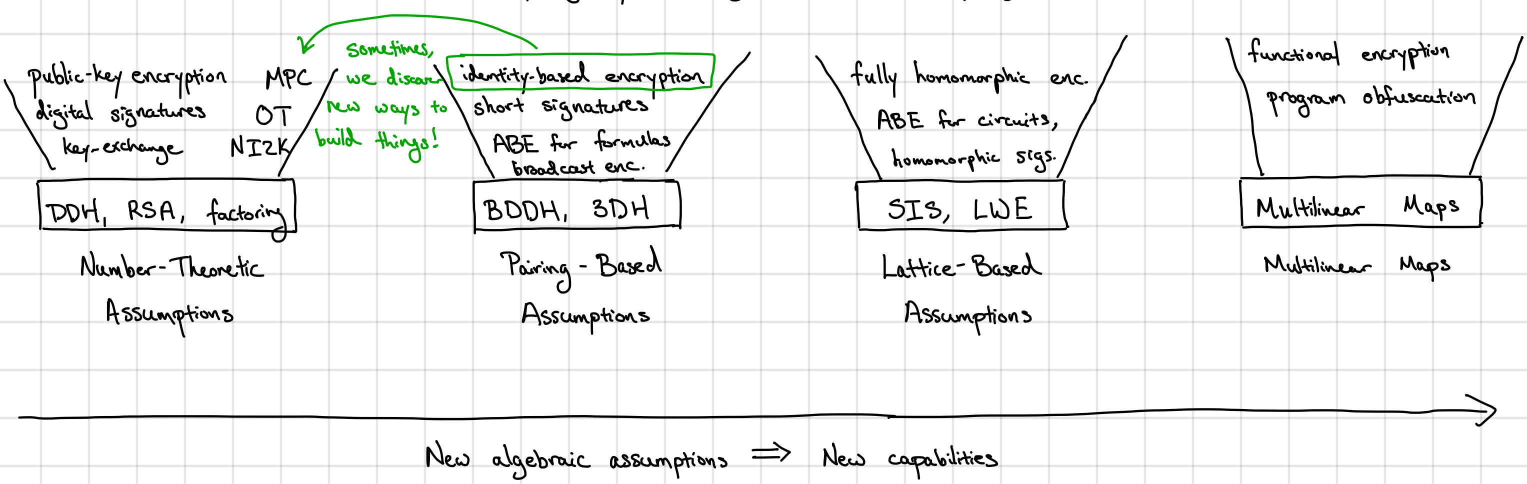
These improvements are closely related to a notion called program obfuscation — one of the most powerful forms of cryptography.

"Ideal" obfuscation: given a program P, Obf(P) implements the same behavior as P, but provides no more information about P than just having black-box access to P

↳ Means that programs cannot be reverse-engineered (unless possible to do just with input/output behavior), allows hiding secrets in software

↳ Extremely powerful notion, implies essentially all of cryptography, but very hard to construct [some notions are even known to be impossible!]

A bird's eye view of the development of cryptography (through the lens of cryptographic hardness):




Many interesting problems still remain ⟹ many opportunities to do research in cryptography!