LaTeX definitions for Cryptography Primitives

Collect definitions of cryptographic primitives so that it is easy to copy and use while writing papers. I will try as far as possible to stick to the macros defined in cryptocode.

Contents

Chapter 1

Notation

We shall denote by $\mathsf{Out}_A\langle A(a), B(b)\rangle$ the output of party A on execution of the protocol between A with input a, and B with input b. By $\mathsf{View}_A\langle A(a), B(b)\rangle$, we denote the view of party A consisting of the protocol transcript along with its random tape.

Chapter 2

Probability

2.1 Computational Indistinguishability

Definition 1 (Computational Indistinguishability). *Two ensembles* $X = \{X_\alpha\}_{\alpha \in S}$ *and* $Y = \{Y_\alpha\}_{\alpha \in S}$ *are said to be computationally indistinguishable, denoted by* $X \approx_c Y$*, if for every non-uniform* PPT *distinguisher* $\mathcal{D}$*, every polynomial* p*, all sufficiently large* λ *and every* $\alpha \in \{0,1\}^{\mathsf{poly}(\lambda)} \cap S$

$$\left| \Pr\left[\mathcal{D}(1^\lambda, X_\alpha) = 1\right] - \Pr\left[\mathcal{D}(1^\lambda, Y_\alpha) = 1\right] \right| < \frac{1}{\mathsf{p}(\lambda)} \ ,$$

where the probability are taken over the samples of X_α*,* Y_α *and coin tosses of* $\mathcal{D}$*.*

2.2 Statistical Indistinguishability

Definition 2 (Statistical Indistinguishability). *Two ensembles* $X = \{X_\alpha\}_{\alpha \in S}$ *and* $Y = \{Y_\alpha\}_{\alpha \in S}$ *are said to be statistically indistinguishable, denoted by* $X \approx_s Y$*, if for every polynomial* p*, all sufficiently large* λ *and every* $\alpha \in \{0,1\}^{\mathsf{poly}(\lambda)} \cap S$

$$\Delta(X_\alpha, Y_\alpha) < \frac{1}{\mathsf{p}(\lambda)} \ ,$$

where $\Delta(X_\alpha, Y_\alpha)$ *corresponds to the statistical distance between* X_α *and* Y_α*.*

Chapter 3

Symmetric Key Primitives

3.1 One-way Function

Definition 3 (One-way Function). *A function $f : \{0,1\}^* \mapsto \{0,1\}^*$ is a one way function if it satisfies the following two conditions:*

1. Easy to compute*: There is a* PPT *algorithm C s.t. $\forall x \in \{0,1\}^*$,*

$$\Pr[r \leftarrow\!\!\$\ \{0,1\}^m \ : \ C(x;r) = f(x)] = 1.$$

2. Hard to invert*: For every non-uniform* PPT *adversary $\mathcal{A}$,*

$$\Pr\big[f(\widetilde{x}) = f(x) \ : \ x \leftarrow\!\!\$\ \{0,1\}^\lambda, \widetilde{x} \leftarrow \mathcal{A}(1^\lambda, f(x))\big] \leq \mathsf{negl}(\lambda)$$

3.2 Pseudorandom Generators

Definition 4 (Psedudorandom Generators). *A deterministic function* PRG : $\{0,1\}^\lambda \to \{0,1\}^{p(\lambda)}$ *is called a pseudorandom generator (PRG) if:*

1. *(efficiency):* PRG *can be computed in polynomial time,*
2. *(expansion): $p(\lambda) > \lambda$,*
3. $\big\{x \leftarrow\!\!\$\ \{0,1\}^\lambda : \mathsf{PRG}(x)\big\} \approx_c \big\{U_{p(\lambda)}\big\}$ *, where $U_{p(\lambda)}$ is the uniform distribution over $p(\lambda)$ bits.*

3.3 Non-interactive Commitment Schemes

We define below bit commitment schemes

Definition 5 (Non-interactive Bit Commitment Schemes). *A polynomial time computable function:* com : $\{0,1\} \times \{0,1\}^\lambda \mapsto \{0,1\}^{\ell(\lambda)}$ *is a bit commitment if it satisfies the properties below:*

Binding: *For any $r, r' \in \{0,1\}^\lambda, b, b' \in \{0,1\}$, if $\mathsf{com}(b;r) = \mathsf{com}(b';r')$ then $b = b'$.*

Computational Hiding: *The following holds:*

$$\left\{\mathsf{com}(0) : r \leftarrow\!\$\ \{0,1\}^\lambda\right\} \approx_c \left\{\mathsf{com}(1; r) : r \leftarrow\!\$\ \{0,1\}^\lambda\right\} \ .$$

where computational indistinguishability is with respect to arbitrary non-uniform PPT *distinguisher.*

3.4 Signature Scheme

Definition 6. *An signature scheme consists of three polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$.

- Gen *is* PPT *algorithm that takes as input* 1^λ *and generates a key and verification key.* $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Gen}(1^\lambda)$.
- Sign *is a* PPT *algorithm that computes the signature on a message* m. $\sigma := \mathsf{Sign}(\mathsf{sk}, m)$.
- Verify *is a deterministic algorithm verifies the signature using the verification key.* $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ *returns 0 or 1.*

A signature scheme that is existentially unforgeable against chosen message attacks if the following hold.

Correctness *For every message* $m \in \mathcal{M}$ *(message space),*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 \ : \ (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\right] = 1$$

Security *For any* PPT *adversary* $\mathcal{A}$

$$\Pr\left[\begin{array}{l} \mathcal{A} \textit{ did not query } m \\ \mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 \end{array} \ : \ \begin{array}{l} (\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}(\mathsf{vk}) \end{array}\right] < \mathsf{negl}(\lambda)$$

where $\mathcal{A}^{\mathsf{Sign}(\mathsf{sk}, \cdot)}$ *indicates that* $\mathcal{A}$ *has access to an oracle that returns the signature on the queried message* m.

Chapter 4

Public Key Primitives

Chapter 5

Proofs

5.1 Non-interactive Witness Indistinguishability (NIWI)

Definition 7. *A non-interactive witness-indistinguishable proof system* $\mathsf{NIWI} = (\mathsf{Prove}, \mathsf{Verify})$ *for an* NP *relation* $R_{\mathcal{L}}$ *consists of two polynomial-time algorithms:*

- *a probabilistic prover* $\mathsf{Prove}(x, w, 1^{\lambda})$ *that given an instance* x*, witness* w*, and security parameter* 1^{λ}*, produces a proof* π*.*
- *a deterministic verifier* $\mathsf{Verify}(x, \pi)$ *that verifies the proof.*

We make the following requirements:

Completeness *for every* $\lambda \in \mathbb{N}, (x, w) \in R_{\mathcal{L}}$*,*

$$\Pr\left[\mathsf{Verify}(x, \pi) = 1 \,:\, \pi \leftarrow \mathsf{Prove}(x, w, 1^{\lambda})\right] = 1$$

Soundness *for every* $x \notin \mathcal{L}$ *and* $\pi \in \{0,1\}^*$*,*

$$\mathsf{Verify}(x, \pi) = 0 \ .$$

Witness Indistinguishability *It holds that*

$$\left\{\mathsf{Prove}(x, w_0, 1^{\lambda})\right\}_{\substack{\lambda, x, \\ w_0, w_1}} \approx_c \left\{\mathsf{Prove}(x, w_1, 1^{\lambda})\right\}_{\substack{\lambda, x, \\ w_0, w_1}} ,$$

where $\lambda \in \mathbb{N}, x \in \{0,1\}^{\lambda}, w_0, w_1 \in R_{\mathcal{L}}(x)$.

5.2 Interactive Proof

Definition 8. *An interactive protocol* (P, V) *between a polynomial time prover* P *and* PPT *verifier* V*, for a language* $\mathcal{L} \in \mathsf{NP}$ *is an interactive proof (resp. argument) if the following holds.*

Completeness: *For every $x \in \mathcal{L}$,*

$$\Pr[\mathsf{Out_V}\langle \mathsf{P}(x,w), \mathsf{V}(x)\rangle = 1] = 1 \ .$$

Soundness (resp. computational soundness): *For any non-uniform (resp.* PPT*)* P^**, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *and* $x \in \{0,1\}^\lambda \setminus \mathcal{L}$*,*

$$\Pr[\mathsf{Out_V}\langle \mathsf{P}^*, \mathsf{V}(x)\rangle = 1] \leq \mathsf{negl}(\lambda) \ .$$

5.3 Zero-Knowledge

An interactive proof (resp. argument) (P, V) between a polynomial time prover P and PPT verifier V, for a language $\mathcal{L}$ is a zero knowledge proof (resp. argument) if the following holds.

Definition 9 (*GMR*[GMR85] Zero-knowledge). *An interactive proof (resp. argument)* (P, V) *between a polynomial time prover* P *and* PPT *verifier* V*, for a language* $\mathcal{L}$ *is a GMR zero knowledge proof (resp. argument) if the following holds. For every* PPT *verifier* V^**, there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{V}^*}$*, such that*

$$\left\{\mathsf{View}_{\mathsf{V}^*}\langle \mathsf{P}(x,w), \mathsf{V}^{*(x)}\rangle\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)}} \approx_c \left\{\mathsf{Sim}_{\mathsf{V}^*}(x)\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)}} \ .$$

Definition 10 (*Auxiliary-input* Zero-knowledge). *An interactive proof (resp. argument)* (P, V) *between a polynomial time prover* P *and* PPT *verifier* V*, for a language* $\mathcal{L}$ *is an auxiliary-input zero knowledge proof (resp. argument) if the following holds. For every* PPT *verifier* V^**, there exists a* PPT *simulator* $\mathsf{Sim}_{\mathsf{V}^*}$*, such that*

$$\left\{\mathsf{View}_{\mathsf{V}^*}\langle \mathsf{P}(x,w), \mathsf{V}^*(x,y)\rangle\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)\\ y\in\{0,1\}^*}} \approx_c \left\{\mathsf{Sim}_{\mathsf{V}^*}(x,y,1^t)\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)\\ y\in\{0,1\}^*}} \ .$$

Definition 11 (*Universal-simulation* Zero-knowledge). *An interactive proof (resp. argument)* (P, V) *between a polynomial time prover* P *and* PPT *verifier* V*, for a language* $\mathcal{L}$ *is a universal-simulation zero knowledge proof (resp. argument) if the following holds. There exists a* PPT *simulator* Sim*, such that for every* PPT *verifier* V^* *of running time at most* $t(\lambda)$*,*

$$\left\{\mathsf{View}_{\mathsf{V}^*}\langle \mathsf{P}(x,w), \mathsf{V}^*(x)\rangle\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)}} \approx_c \left\{\mathsf{Sim}(\mathsf{V}^*, 1^t, x)\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)}} \ .$$

Definition 12 (*Black-box-simulation* Zero-knowledge). *An interactive proof (resp. argument)* (P, V) *between a polynomial time prover* P *and* PPT *verifier* V*, for a language* $\mathcal{L}$ *is a black-box-simulation zero knowledge proof (resp. argument) if the following holds. There exists a* PPT *simulator* Sim*, such that for every* PPT *verifier* V^**,*

$$\left\{\mathsf{View}_{\mathsf{V}^*}\langle \mathsf{P}(x,w), \mathsf{V}^*\rangle\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)}} \approx_c \left\{\mathsf{Sim}^{\mathsf{V}^*}(x)\right\}_{\substack{\lambda\in\mathbb{N},\\ x\in\mathcal{L}\cap\{0,1\}^\lambda,\\ w\in R_{\mathcal{L}}(x)}} \ .$$

Chapter 6

Secure Computation

Chapter 7

Obfuscation

7.1 Indistinguishability Obfuscator for Turing Machines

Definition 13 (Indistinguishability Obfuscator for Turing Machines). *A succinct indistinguishability obfuscator for Turing machines consists of a PPT machine* iOM *that works as follows:*

- iOM *takes as input the security parameter* 1^λ*, the Turing machine* M *to obfuscate, an input length* n*, and time bound* t*.*
- iOM *outputs a Turing machine* $\widetilde{\mathsf{M}}$ *which is an obfuscation of* M *corresponding to input length* n *and time bound* t*.* $\widetilde{\mathsf{M}}$ *takes as input* $x \in \{0,1\}^n$*.*

The scheme should satisfy the following requirements:

Correctness *For all* $\lambda \in \mathbb{N}$*, for all* $\mathsf{M} \in \mathcal{M}_\lambda$*, for all inputs* $x \in \{0,1\}^n$*, time bounds* t' *such that* $t' \leq t$*, let* y *be the output of* $\mathsf{M}(x)$ *after at most* t *steps, then*

$$\Pr\left[\widetilde{\mathsf{M}}(x) = y \;:\; \widetilde{\mathsf{M}} \leftarrow \mathsf{iOM}(1^\lambda, 1^n, 1^{\log t}, \mathsf{M})\right] = 1 \;\;.$$

Security *It holds that*

$$\left\{\mathsf{iOM}(1^\lambda, 1^n, 1^{\log t}, \mathsf{M}_0)\right\}_{\substack{\lambda,t,n,\\ \mathsf{M}_0,\mathsf{M}_1}} \approx_c \left\{\mathsf{iOM}(1^\lambda, 1^n, 1^{\log t}, \mathsf{M}_1)\right\}_{\substack{\lambda,t,n,\\ \mathsf{M}_0,\mathsf{M}_1}} \;\;,$$

where $\lambda \in \mathbb{N}$*,* $n \leq t \leq 2^\lambda$*, and* $\mathsf{M}_0, \mathsf{M}_1$ *are any pair of machines of the same size such that for any input* $x \in \{0,1\}^n$ *both halt after the same number of steps with the same output.*

Efficiency and Succinctness *We require that the running time of* iOM *and the length of its output, namely the obfuscated machine* $\widetilde{\mathsf{M}}$*, is* $\mathsf{poly}(|\mathsf{M}|, \log t, n, \lambda)$*. We also require that the running time* $\tilde{t}_x$ *of* $\widetilde{\mathsf{M}}(x)$ *is* $\mathsf{poly}(t_x, |\mathsf{M}|, n, \lambda)$*, where* t_x *is the running time of* $\mathsf{M}(x)$*.*

7.2 Witness Encryption

Definition 14. *A witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *for an* NP *language* $\mathcal{L}$*, with corresponding witness relation* $R_\mathcal{L}$*, consists of the following two polynomial-time algorithms:*

Encryption. *The probabilistic algorithm* $\mathsf{Enc}(1^\lambda, x, m)$ *takes as input a security parameter* 1^λ*, a string* $x \in \{0,1\}^*$*, and a message* $m \in \{0,1\}$*. It outputs a ciphertext* ct.

Decryption. *The algorithm* $\mathsf{Dec}(\mathsf{ct}, w)$ *takes as input a ciphertext* ct*, a string* $w \in \{0,1\}^*$*. It outputs either a message* $m \in \{0,1\}$.

The above algorithms satisfy the following conditions:

– ***Correctness.*** *For any security parameter* λ*, for any* $m \in \{0,1\}$*, and for any* $(x, w) \in R_{\mathcal{L}}$*, we have that*

$$\Pr\left[\mathsf{Dec}(\mathsf{ct}, w) = m \,:\, \mathsf{ct} \leftarrow \mathsf{Enc}(1^\lambda, x, m)\right] = 1 \ .$$

– ***Security.*** *For any non-uniform* PPT*adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for any* $\lambda \in \mathbb{N}$*, and any* $x \notin \mathcal{L}$*, we have that*

$$\left\{\mathsf{Enc}(1^\lambda, x, 0)\right\}_{\lambda \in \mathbb{N}, x \notin \mathcal{L}} \approx_c \left\{\mathsf{Enc}(1^\lambda, x, 1)\right\}_{\lambda \in \mathbb{N}, x \notin \mathcal{L}} \ .$$

Bibliography

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th Annual ACM Symposium on Theory of Computing*, pages 291–304, Providence, RI, USA, May 6–8, 1985. ACM Press.

Index