DefCrypt

Defining Cryptography

Mihir Bellare
and
The Students of 209B

June 5, 2017

Abstract

The project aims to create a repository for cryptographic definitions. We seek a comprehensive, unified rendition of cryptographic definitions.

Latex files can be downloaded and used. For definitions that are novel, please check with us first. Use your judgement with regard to citation.

MOTIVATION. Definitions allow us to formalize and capture security goals. They are the basis for reduction-based (also called provable security) cryptography. They are also valuable in cryptanalysis as targets for attack. The proliferation of definitions, written in different style, with different degrees of precision, can make the academic literature hard to navigate. Our intent is a more structured approach. Our definitions are proof-friendly. They make it easier to write reduction-based proofs by naming all oracles in games. When working on a problem or writing a paper, one can feel confusion, or that it is not clear how to write something precisely and in detail. The fault often lies with the definitions. We think ours will help.

Contents

1 Repository Organization **3**

2 Conventions **4**

3 Examples **5**

4 Commands **6**
4.1 Useful Commands 6
4.1.1 Theorem, Lemma, etc. Environments 6
4.1.2 Math Environments 6
4.1.3 Section Heading Stuff 6
4.1.4 Figure Stuff 7
4.1.5 General Math 7
4.1.6 Set Stuff 7
4.1.7 Probability/Expected Value 7
4.1.8 Crypto Specific 7
4.2 Less Important 11
4.2.1 Something with fonts ??? 11
4.2.2 ??? 11
4.2.3 List environments 11
4.2.4 Spacing Something ??? 11
4.2.5 Math Relations/Operations with spacing/size stuff ??? 11
4.2.6 Fonts ??? 11
4.2.7 Colors 12
4.2.8 Author Notes 12
4.2.9 Symbols for math-y phrases ??? 12
4.3 Acknowledgement 12

Chapter 1

Repository Organization

The current organization of the repository was designed aiming for modularity for ease of git coordination and eventual Wiki uploading.

Each type of cryptographic primitive will have its own folder within the "/primitives" folder. For example, functions families are discussed in the "/primitives/ff" folder. Each of these subfolders will have a main tex file which can be compiled separately from the global repository tex file. They will additionally have subfolders for syntax and security notions. Each security notion will individually correspond to an individual tex file.

Citation for papers in cryptography can be found in the cryptobib folder (make sure you clone from git with the –recursive flag to include this folder).

Chapter 2

Conventions

In this section we might add some conventions of how to write content.

Try to pattern match the latex conventions of other writing in the repository so we maintain as much uniformity of style as possible.

Chapter 3

Examples

In this section we might add some examples of how to use the LaTeX commands.

Chapter 4

Commands

4.1 Useful Commands

The organization of this section follows the organization of "defrypt.sty", skipping over commands I consider less important which are included in Section 4.2. The section titles correspond closely to comments used to organize the "decrypt.sty" file itself.

4.1.1 Theorem, Lemma, etc. Environments

This section defines environments for stating theorems, lemmas, corollary, propositions, definitions, claims, remarks, examples, and notes.

Additionally, it defines commands which should always be used when referencing a label. The convention for naming labels is to first use the abbreviation for the type of thing corresponding to the reference command included below, followed by then the relevant type of primitive if any, followed by a clear name (each of these should be separated by a hyphen). For example, a figure showing the PRF game in the function family section could be labelled with \label{fig-ff-prfgame}.

Command	**Description**	**Result**
\secref{sec-one}	Natural numbers	Section 1
\apref{ap-one}	Integers	Appendix 1
\thref{th-one}	Reals	Theorem 1
\defref	A tuple	Definition 1
\corref	Parentheses	Corollary 1
\lemref	Size of a set	Lemma 1
\clref	Length of a string	Claim **??**
\propref	The empty string	Proposition 1
\figref	Concatenation	Fig. 1
\eqref	Cartesian product	Equation (1)

4.1.2 Math Environments

This section defines the math environments "newmath", "neweqnarray", and "new equation".

4.1.3 Section Heading Stuff

The commands \heading and \noskipheading are used to give a heading to a small chunk of text.

4.1.4 Figure Stuff

This sections defines the commands \oneCol, \twoColsNoDivide, \twoCols, ... which may be useful for creating minipages to contain pseudocode in figures. Currently they do not seem to interact nicely with Cryptocode so it is unclear if we will use these commands.

These commands tend to take several inputs the first few of which specify the widths of the different minipages as a fraction of the total page length. For example, .33 gives a minipage approximately a third of the size of a page.

4.1.5 General Math

Command	Description	Result
\N	Natural numbers	$\mathbb{N}$
\Z	Integers	$\mathbb{Z}$
\R	Reals	$\mathbb{R}$
\tuple{x,y}	A tuple	$\langle x, y \rangle$
\prn{x,y}	Parentheses	(x, y)
\setsize{S}	Size of a set	$\lvert S \rvert$
\len{x}	Length of a string	$\lvert x \rvert$
\emptystring	The empty string	ε
x\concat y	Concatenation	$x \,\|\| y$
\cross	Cartesian product	$\times$

4.1.6 Set Stuff

Command	Description	Result
\sett{a,b,c}	A set	$\{a, b, c\}$
\set{a}{b>a}	Set	$\{\, a \,:\, b > a \,\}$
\setX	A set called X	X

4.1.7 Probability/Expected Value

Command	Description	Result
\Prob{\bad}	Probability	$\Pr[\,\mathsf{bad}\,]$
\condProb{\bad}{b=1}	Conditional probability	$\Pr[\,\mathsf{bad} \mid b = 1\,]$
\Var{X}	Variance	$\mathbf{Var}[X]$
\E	Expected value	$\mathbf{E}$
\EE{X}	Expected value	$\mathbf{E}[X]$
\EEE{b\getsr\bits}{X}	Expected value	$\mathbf{E}_{b \leftarrow\$ \{0,1\}}[X]$
\condE{X}{b=1}	Conditional expected value	$\mathbf{E}[\,X \mid b = 1\,]$

4.1.8 Crypto Specific

This section contains some commands which are particularly useful for cryptography.

Command	Description	Result
\getsr	Random sampling	$\leftarrow\!\!\!\!\!^{\$}$
\bits	Bits	$\{0,1\}$
\xor	Exclusive or	$\oplus$
\secIn	Security parameter in unary	1^λ
\secParam	Security parameter in binary	λ
\msg	A message	M
\ciph	A ciphertext	C
\nonce	A nonce	N
\iv	An initialization vector	N

Adversaries

Adversaries all use the font \advfont.

Command	Description	Result
\advA	Adversary A	A
\advB	Adversary B	B

Schemes

New schemes should be defined to use the font \schemefont. Some schemes currently include:

Command	Description	Result
\HF	A family of functions	H
\FF	A family of functions	F
\GF	A family of functions	G
\oGF	A family of functions	$\overline{\mathsf{G}}$
\OO	An obfuscator	O
\SE	A symmetric encryption scheme	SE

Dot Extensions

We use what we call the dot syntax. For example a function family F specifies its keyspace $\mathsf{F.Keys}$, domain $\mathsf{F.Dom}$ and range $\mathsf{F.Rng}$ as shown, so that $\mathsf{F}\colon \mathsf{F.Keys} \times \mathsf{F.Dom} \to \mathsf{F.Rng}$. This means that, once we write F, we can invoke the associated sets directly via the dot extensions.

Traditionally, the syntax of a primitive is a tuple, like a PKE scheme is the triple of key generation, encryption and decryption algorithms. Now, the scheme is a single object with the algorithms specified after the dots via standard extensions.

It makes notation compact. It also allows expansion. If you want to refer to some unconventional element (for example, length of the randomness), add a dot extension. It also moves us closer to a programming language rendition, which is valuable both pedagogically (PlayCrypt) and pragmatically.

New dot extensions will use the font \schalg. The commands for an extension will take as input the scheme it is extending and use the same capitalization as the ultimate output. Algorithms and sets will them to have a capitalized first letter.

Some extensions currently include:

Command	Description	Result
\Dom{\FF}	Domain	$\mathsf{F.Dom}$
\Rng{\FF}	Range	$\mathsf{F.Rng}$
\Keys{\FF}	Keyspace	$\mathsf{F.Keys}$
\NS{\SE}	Nonce space	$\mathsf{SE.NS}$
\Kg{\FF}	Key generation	$\mathsf{F.Kg}$
\Ev{\FF}	Evaluation	$\mathsf{F.Ev}$
\Inv{\FF}	Inverse	$\mathsf{F.In}$
\Enc{\SE}	Encryption	$\mathsf{SE.Enc}$
\Dec{\SE}	Decryption	$\mathsf{SE.Dec}$
\Obf{\OO}	Obfuscation	$\mathsf{O.Obf}$
\Samp{\sampS}	Sampling	$S.\mathsf{Samp}$
\Is	Input sampling	$\mathsf{F.Is}$

Length functions will tend to be all lowercase:

Command	Description	Result
\ol{\FF}	Output length	$\mathsf{F.ol}$
\il{\FF}	Input length	$\mathsf{F.il}$
\kl{\FF}	Key length	$\mathsf{F.kl}$
\cl{\SE}	Ciphertext length	$\mathsf{SE.cl}$
\bl{\FF}	Block length	$\mathsf{F.bl}$
\el{\OO}	Expansion length	$\mathsf{O.el}$

Pseudocode Commands

In addition to pseudo commands specified by the Cryptocode package we have the following commands for writing pseudocode predefined.

Command	Description	Result
\true	True boolean	true
\false	False boolean	false
\bad	Flag set to true when something "bad" occurs	bad
a\ind b	An indent used in code	$a \quad b$

Games

Games are our preferred method of writing security notion. They allow a precise formalization that is intended to be proof-friendly and easily understood. They will be written as a single G using the font \gamefont. The game will have a superscript giving its name and the command will take as input a subscript. The convention for game commands with be a single 'g' followed by the name of the security game. Each new security notation should specify a game.

Currently included games include:

Command	Description	Result
\gOW{\FF,\advA}	One way function game	$\mathbf{G}^{\mathrm{ow}}_{\mathsf{F},A}$
\gPRF{\FF,\advA}	Pseudorandom function game	$\mathbf{G}^{\mathrm{prf}}_{\mathsf{F},A}$
\gROR{\SE,\advA}	Real or random game	$\mathbf{G}^{\mathrm{ror}}_{\mathsf{SE},A}$
\gmuIND{\SE}	Multi-user indistinguishability game	$\mathbf{G}^{\text{mu-ind}}_{\mathsf{SE}}$
\gmuKR{\SE}	Multi-user key recovery game	$\mathbf{G}^{\text{mu-kr}}_{\mathsf{SE}}$
\gIO{\OO,\advA}	Indistinguishability obfuscation game	$\mathbf{G}^{\mathrm{io}}_{\mathsf{O},A}$

Advantages

We use the convention of writing security definitions in terms of an advantage function which measures how well an adversary breaks the security of a primitive. They allow a convenient modular statements of theorems and encourage definitions that are amenable to being considered in both the asymptotic and concrete security regimes. Each advantage function will include its name as a superscript. The command for it will take two inputs, a subscript and an input to the function. In the asymptotic section, the input to the advantage functions should be the security parameter in binary, in the concrete setting it will be the adversary. The convention for advantage commands is the name of the security notion in lowercase followed by "Adv". Every game should have a advantage function associated with it.

Currently included advantage functions include:

Command	**Description**	**Result**
\owAdv{\FF,\advA}{\secParam}	One way function advantage	$\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{F},A}(\lambda)$
\prfAdv{\FF,\advA}{\secParam}	Pseudorandom function advantage	$\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F},A}(\lambda)$
\rorAdv{\SE}{\advA}	Real or random advantage	$\mathbf{Adv}^{\mathrm{ind\$}}_{\mathsf{SE}}(A)$
\muindAdv{\SE}{\advA}	Multi-user indistinguishability advantage	$\mathbf{Adv}^{\mathrm{mu\text{-}ind}}_{\mathsf{SE}}(A)$
\mukrAdv{\SE}{\advA}	Multi-user key recovery advantage	$\mathbf{Adv}^{\mathrm{mu\text{-}kr}}_{\mathsf{SE}}(A)$
\ioAdv{\OO,\advA,\sampS}{\secParam}	Indistinguishability obfuscation advantage	$\mathbf{Adv}^{\mathrm{io}}_{\mathsf{O},A,S}(\lambda)$

Security Sets

In the asymptotic setting, we will define sets of all of the security schemes for a particular security notation. These sets often allow precise and succinct theorem statements relating different security notions. The naming convention for commands is to have "set" followed by the name of the security notion in uppercase. Every asymptotic security notion should have a set associated with it.

Currently included sets include:

Command	**Description**	**Result**
\setOW	One way function set	**OW**
\setPRF	Pseudorandom function set	**PRF**
\setROR	Real or random set	**ROR**
\setmuIND	Multi-user indistinguishability set	**MU-IND**
\setmuKR	Multi-user key recovery set	**MU-KR**
\setIO	Indistinguishability obfuscation set	**IO**

Oracles

Games often give an adversary access to an oracle. We have various predefined oracles names. New oracles will use the font \procfont. We will use the convention that the commands for oracles end in a capital 'O'.

Currently included oracles include:

Command	Description	Result
\EncO	Encryption oracle	Enc
\VfO	Verification oracle	Vf
\NewO	New oracle	New
\LRO	Left-or-right oracle	LR
\RoRO	Real-or-random oracle	RoR
\FnO	Function oracle	Fn

Keys

Schemes often need to use keys. We have various predefined macros for them.

Command	Description	Result
\key	Generic key	K
\fkey	F key	K
\nkey	N key	K
\pkey	Public key	pk
\skey	Secret key	sk

4.2 Less Important

This section includes various commands that I (Joseph) consider less important to know.

4.2.1 Something with fonts ???

I dunno, something with fonts.

4.2.2 ???

?

4.2.3 List environments

This section defines some list environments.

4.2.4 Spacing Something ???

This section defines some lengths.

4.2.5 Math Relations/Operations with spacing/size stuff ???

The commands \leqq, \eqq, \geqq, \equivv, \seq, \sminus, \splus, and \Colon have uses. I guess.

4.2.6 Fonts ???

This section defines several fonts. I do not believe any of them are currently in use, they may be removed.

4.2.7 Colors

This section defines the commands \red, \green, \blue, and \cyan. These commands can be useful for making text easy to find while debugging, but should be avoided in finalized writing.

4.2.8 Author Notes

The command \authnote{} currently does not work.

4.2.9 Symbols for math-y phrases ???

I do not have an explanation for \then, \andthen, \suchthatt, or \suchthat. Perhaps they are reasonable to remove?

4.3 Acknowledgement

Some formatting of this LaTeX file was borrowed from Cryptocode by Arno Mittelbach.