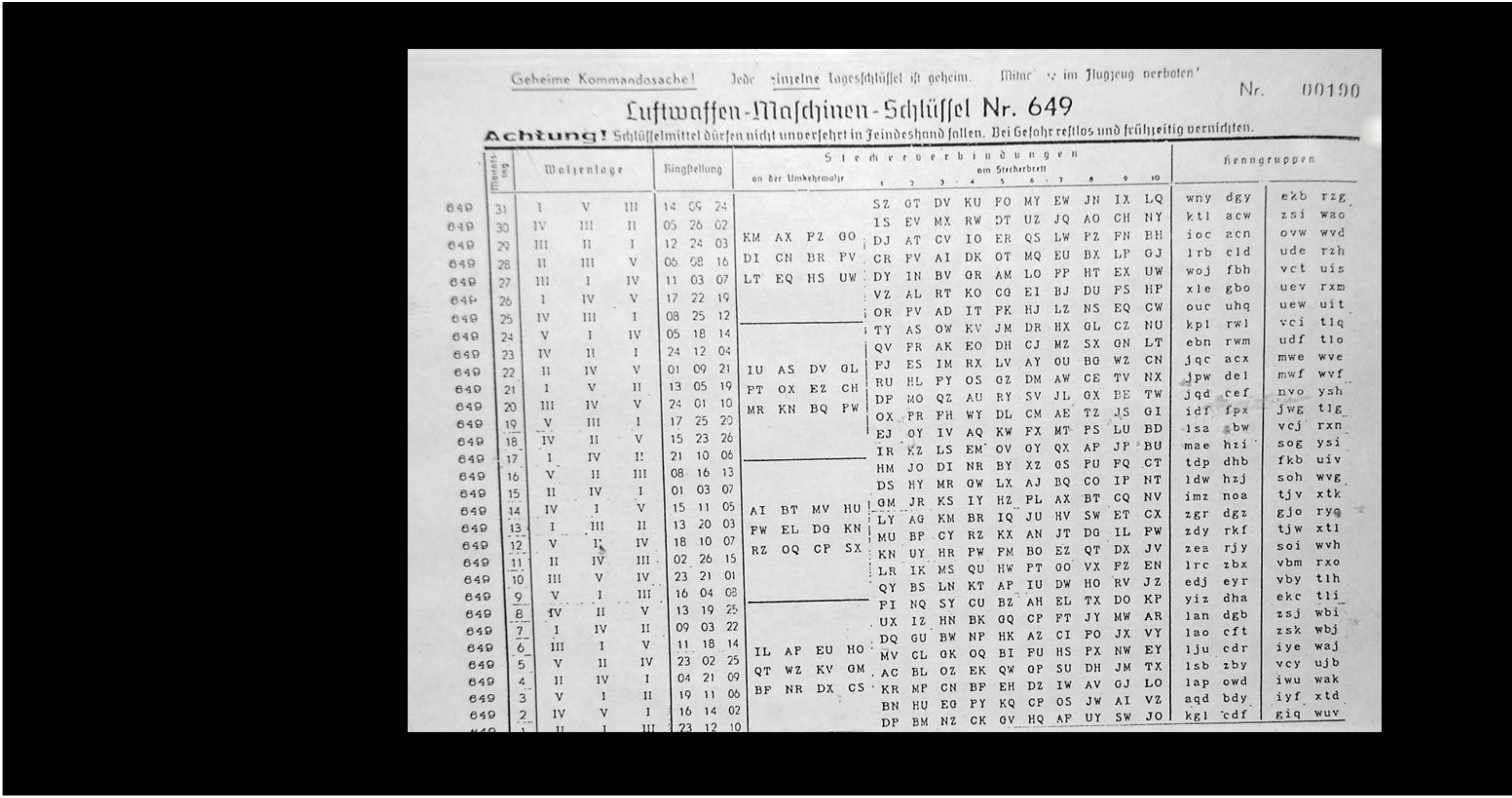
Example of applied cryptography used in SDDR: Diffie Hellman key exchange

Diffie-Hellman Key Exchange

Whittfield Diffie and Martin Hellman are called the inventors of Public Key Cryptography. Diffie-Hellman Key Exchange is the first Public Key Algorithm published in 1976.



Geheime Kommandosache! Jede einzelne Tagesschlüssel ist geheim. Mitnahme im Flugzeug verboten! Nr. 00190

Luftwaffen-Maschinen-Schlüssel Nr. 649

Achtung! Schlüsselmittel dürfen nicht unversehrt in Feindeshand fallen. Bei Gefahr restlos und frühzeitig vernichten.

	Monatstag	Walzenlage			Ringstellung			Steckerverbindungen an der Umkehrwalze	am Steckerbrett 1	2	3	4	5	6	7	8	9	10	Kenngruppen			
649	31	I	V	III	14	09	24		SZ	GT	DV	KU	FO	MY	EW	JN	IX	LQ	wny	dgy	ekb	rzg
649	30	IV	III	II	05	26	02		IS	EV	MX	RW	DT	UZ	JQ	AO	CH	NY	ktl	acw	zsi	wao
649	29	III	II	I	12	24	03	KM AX PZ GO	DJ	AT	CV	IO	ER	QS	LW	PZ	FN	BH	ioc	acn	ovw	wvd
649	28	II	III	V	06	08	16	DI CN BR PV	CR	FV	AI	DK	OT	MQ	EU	BX	LP	GJ	lrb	cld	ude	rzh
649	27	III	I	IV	11	03	07	LT EQ HS UW	DY	IN	BV	GR	AM	LO	PP	HT	EX	UW	woj	fbh	vct	uis
649	26	I	IV	V	17	22	19		VZ	AL	RT	KO	CG	EI	BJ	DU	FS	HP	xle	gbo	uev	rxm
649	25	IV	III	I	08	25	12		OR	PV	AD	IT	PK	HJ	LZ	NS	EQ	CW	ouc	uhq	uew	uit
649	24	V	I	IV	05	18	14		TY	AS	OW	KV	JM	DR	HX	GL	CZ	NU	kpl	rwl	vci	tlq
649	23	IV	II	I	24	12	04		QV	FR	AK	EO	DH	CJ	MZ	SX	GN	LT	ebn	rwm	udf	tlo
649	22	II	IV	V	01	09	21	IU AS DV GL	FJ	ES	IM	RX	LV	AY	OU	BG	WZ	CN	jqc	acx	mwe	wve
649	21	I	V	II	13	05	19	PT OX EZ CH	RU	HL	PY	OS	GZ	DM	AW	CE	TV	NX	jpw	del	mwf	wvf
649	20	III	IV	V	24	01	10	MR KN BQ PW	DF	MO	QZ	AU	RY	SV	JL	GX	BE	TW	jqd	cef	nvo	ysh
649	19	V	III	I	17	25	20		OX	PR	FH	WY	DL	CM	AE	TZ	JS	GI	idf	fpx	jwg	tlg
649	18	IV	II	V	15	23	26		EJ	OY	IV	AQ	KW	FX	MT	PS	LU	BD	lsa	[illegible]	vcj	rxn
649	17	I	IV	II	21	10	06		IR	KZ	LS	EM	OV	GY	QX	AF	JP	BU	mae	hzi	sog	ysi
649	16	V	II	III	08	16	13		HM	JO	DI	NR	BY	XZ	GS	PU	FQ	CT	tdp	dhb	fkb	uiv
649	15	II	IV	I	01	03	07		DS	HY	MR	GW	LX	AJ	BQ	CO	IP	NT	ldw	hzj	soh	wvg
649	14	IV	I	V	15	11	05	AI BT MV HU	GM	JR	KS	IY	HZ	PL	AX	BT	CQ	NV	imz	noa	tjv	xtk
649	13	I	III	II	13	20	03	FW EL DG KN	LY	AG	KM	BR	IQ	JU	HV	SW	ET	CX	zgr	dgz	gjo	ryq
649	12	V	II	IV	18	10	07	RZ OQ CP SX	MU	BP	CY	RZ	KX	AN	JT	DG	IL	FW	zdy	rkf	tjw	xtl
649	11	II	IV	III	02	26	15		KN	UY	HR	PW	FM	BO	EZ	QT	DX	JV	zea	rjy	soi	wvh
649	10	III	V	IV	23	21	01		LR	IK	MS	QU	HW	PT	GO	VX	FZ	EN	lrc	zbx	vbm	rxo
649	9	V	I	III	16	04	08		QY	BS	LN	KT	AP	IU	DW	HO	RV	JZ	edj	eyr	vby	tlh
649	8	IV	II	V	13	19	25		FI	NQ	SY	CU	BZ	AH	EL	TX	DO	KP	yiz	dha	ekc	tli
649	7	I	IV	II	09	03	22		UX	IZ	HN	BK	GQ	CP	FT	JY	MW	AR	lan	dgb	zsj	wbi
649	6	III	I	V	11	18	14	IL AP EU HO	DQ	GU	BW	NP	HK	AZ	CI	PO	JX	VY	lao	cft	zsk	wbj
649	5	V	II	IV	23	02	25	QT WZ KV GM	MV	CL	GK	OQ	BI	FU	HS	PX	NW	EY	lju	cdr	iye	waj
649	4	II	IV	I	04	21	09	BF NR DX CS	AC	BL	OZ	EK	QW	GP	SU	DH	JM	TX	lsb	zby	vcy	ujb
649	3	V	I	II	19	11	06		KR	MP	CN	BF	EH	DZ	IW	AV	GJ	LO	lap	owd	iwu	wak
649	2	IV	V	I	16	14	02		BN	HU	EG	PY	KQ	CP	OS	JW	AI	VZ	aqd	bdy	iyf	xtd
649	1	II	I	III	23	12	10		DP	BM	NZ	CK	GV	HQ	AF	UY	SW	JO	kgl	cdf	giq	wuv

Prior to public key methods like Diffie–Hellman, cryptographic keys had to be transmitted in physical form such as this World War II list of keys for the German Enigma cipher machine.

What is Diffie-Hellman?

- A Public Key Algorithm
- Only for Key Exchange
- Does NOT Encrypt or Decrypt
- Based on Discrete Logarithms
- Widely used in Security Protocols and Commercial Products
- Williamson of Britain's CESG claims to have discovered it several years prior to 1976

Sets, Groups and Fields

- A set is any collection of objects called the elements of the set
- Examples of sets: ***R***, ***Z***, ***Q***
- If we can define an operation on the elements of the set and certain rules are followed then we get other mathematical structures called groups and fields

Groups

- A group is a set G with a custom-defined binary operation $+$ such that:
  - The group is closed under $+$, i.e., for $a, b \in G$:
    - $a + b \in G$
  - The Associative Law holds i.e., for any $a, b, c \in G$:
    - $a + (b + c) = (a + b) + c$
  - There exists an identity element 0, such that
    - $a + 0 = a$
  - For each $a \in G$ there exists an inverse element $-a$ such that
    - $a + (-a) = 0$
- If for all $a, b \in G$: $a + b = b + a$ then the group is called an Abelian or commutative group
- If a group G has a finite number of elements it is called a finite group

More About Group Operations

- + does not necessarily mean normal arithmetic addition
- + just indicates a binary operation which can be custom defined
- The group operation could be denoted as •
- The group notation with + is called the additive notation and the group notation with • is called the multiplicative notation

Fields

- A field is a set F with two custom-defined binary operations + and • such that:
  - The Field is closed under + and •, i.e., for $a, b \in F$:
    - $a + b \in F$ and $a \cdot b \in F$
  - The Associative Law holds i.e., for any $a, b, c \in F$:
    - $a + (b + c) = (a + b) + c$ and $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
  - There exist identity elements 0 and 1, such that
    - $a + 0 = a$ and $a \cdot 1 = a$
  - For each $a \in F$ there exist inverse elements $-a$ and a^{-1} such that
    - $a + (-a) = 0$ and $a \cdot a^{-1} = 1$
- If a field F has a finite number of elements it is called a finite field

Examples of Groups

- Groups
  - Set of real numbers ***R*** under +
  - Set of real numbers ***R*** under *
  - Set of integers ***Z*** under +
  - Set of integers ***Z*** under *?
  - Set of integers modulo a prime number *p* under +
  - Set of integers modulo a prime number *p* under *
  - Set of 3 X 3 matrices under + meaning matrix addition
  - Set of 3 X 3 matrices under * meaning matrix multiplication?
- Fields
  - Set of real numbers ***R*** under + and *
  - Set of integers ***Z*** under + and *
  - Set of integers modulo a prime number *p* under + and *

Generator of Group

- If for $a \in G$, all members of the group can be written in terms of a by applying the group operation * on a a number of times then a is called a generator of the group G
- Examples
  - 2 is a generator of Z^*_{11}
  - 2 and 3 are generator of Z^*_{19}

m	1	2	3	4	5	6	7	8	9	10
2^m mod 11	2	4	8	5	10	9	7	3	6	1

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
2^m mod 19	2	4	8	16	13	7	14	3	6	17	15	11	3	6	12	5	10	1
3^m mod 19	3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1

Primitive Roots

- If $x^n = a$ then a is called the *n-th* root of x
- For any prime number p, if we have a number a such that powers of a mod p generate all the numbers between 1 to $p-1$ then a is called a **Primitive Root** of p.
- In terms of the Group terminology a is the generator element of the multiplicative group of the finite field formed by mod p
- Then for any integer b and a primitive root a of prime number p we can find a unique exponent i such that

$$b = a^i \; mod \; p$$

- The exponent i is referred to as the **discrete logarithm** or **index**, of b for the base a.

Table 7.6 Powers of Integers, Modulo 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Table 7.7 Tables of Discrete Logarithms, Modulo 19

(a) Discrete logarithms to the base 2, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Ind_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(b) Discrete logarithms to the base 3, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Ind_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(c) Discrete logarithms to the base 10, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Ind_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(d) Discrete logarithms to the base 13, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Ind_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(e) Discrete logarithms to the base 14, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Ind_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	14	9

(f) Discrete logarithms to the base 15, modulo 19

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$Ind_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	12	9

Diffie-Hellman Algorithm

- Five Parts
  1. Global Public Elements
  2. User A Key Generation
  3. User B Key Generation
  4. Generation of Secret Key by User A
  5. Generation of Secret Key by User B

Global Public Elements

- q Prime number
- α $\alpha < q$ and α is a primitive root of q
- The global public elements are also sometimes called the domain parameters

User A Key Generation

- Select private X_A $\quad$ $X_A < q$
- Calculate public Y_A $\quad$ $Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

- Select private X_B $X_B < q$
- Calculate public Y_B $Y_B = \alpha^{X_B} \bmod q$

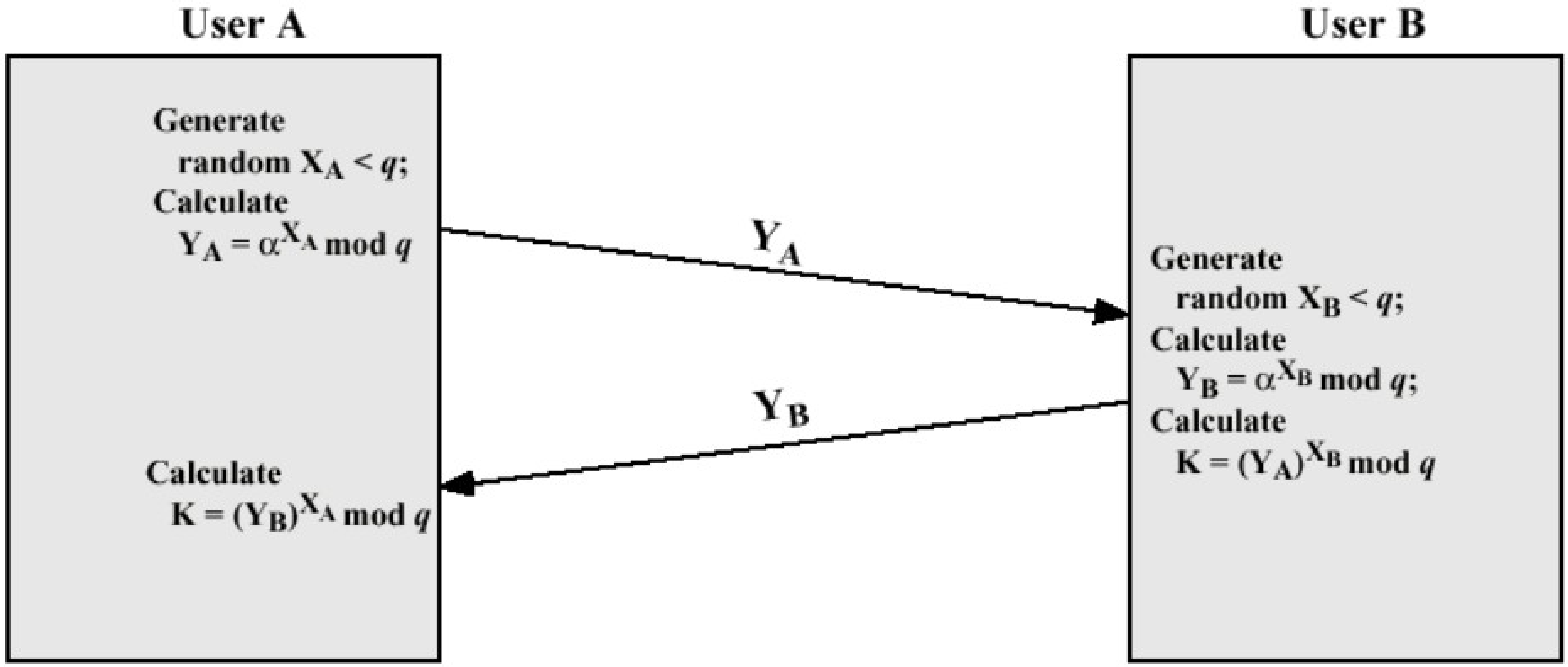
Generation of Secret Key by User A

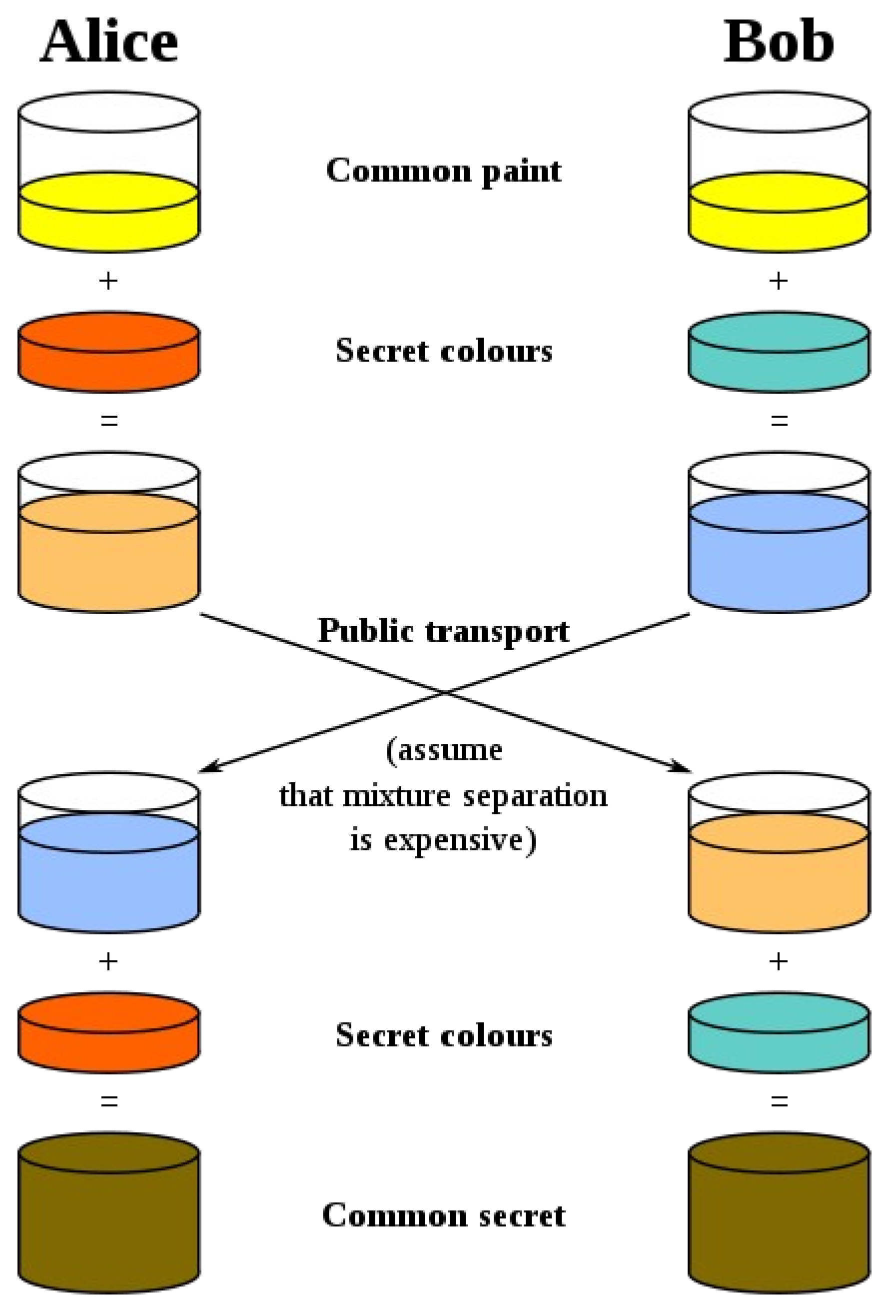
- $K = (Y_B)^{X_A} \bmod q$

Generation of Secret Key by User B

- $K = (Y_A)^{X_B} \bmod q$

Diffie-Hellman Key Exchange




Alice
Bob
Common paint
+
Secret colours
=
Public transport
(assume
that mixture separation
is expensive)
+
Secret colours
=
Common secret

Diffie-Hellman Example

1. Alice and Bob agree on $p = 23$ and $g = 5$.
2. Alice chooses $a = 6$ and sends $5^6 \mod 23 = 8$.
3. Bob chooses $b = 15$ and sends $5^{15} \mod 23 = 19$.
4. Alice computes $19^6 \mod 23 = 2$.
5. Bob computes $8^{15} \mod 23 = 2$.

Then 2 is the shared secret.

Clearly, much larger values of a, b, and p are required. An eavesdropper cannot discover this value even if she knows p and g and can obtain each of the messages.

Why Diffie-Hellman is Secure?

- Opponent has q, α, Y_A and $Y_{\mathbf{B}}$
- To get X_A or X_B the opponent is forced to take a discrete logarithm
- The security of the Diffie-Hellman Key Exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

Diffie-Hellman Security

Suppose p is a prime of around 300 digits, and a and b at least 100 digits each.

Discovering the shared secret given g, p, $g^a \mod p$ and $g^b \mod p$ would take longer than the lifetime of the universe, using the best known algorithm. This is called the *discrete logarithm problem*.

Diffie-Hellman is currently used in many protocols, namely:

- Secure Sockets Layer (SSL)/Transport Layer Security (TLS)
- Secure Shell (SSH)
- Internet Protocol Security (IPSec)
- Public Key Infrastructure (PKI)

Example of Efficient Data Structure used in SDDR: Bloom Filter

Origin and applications

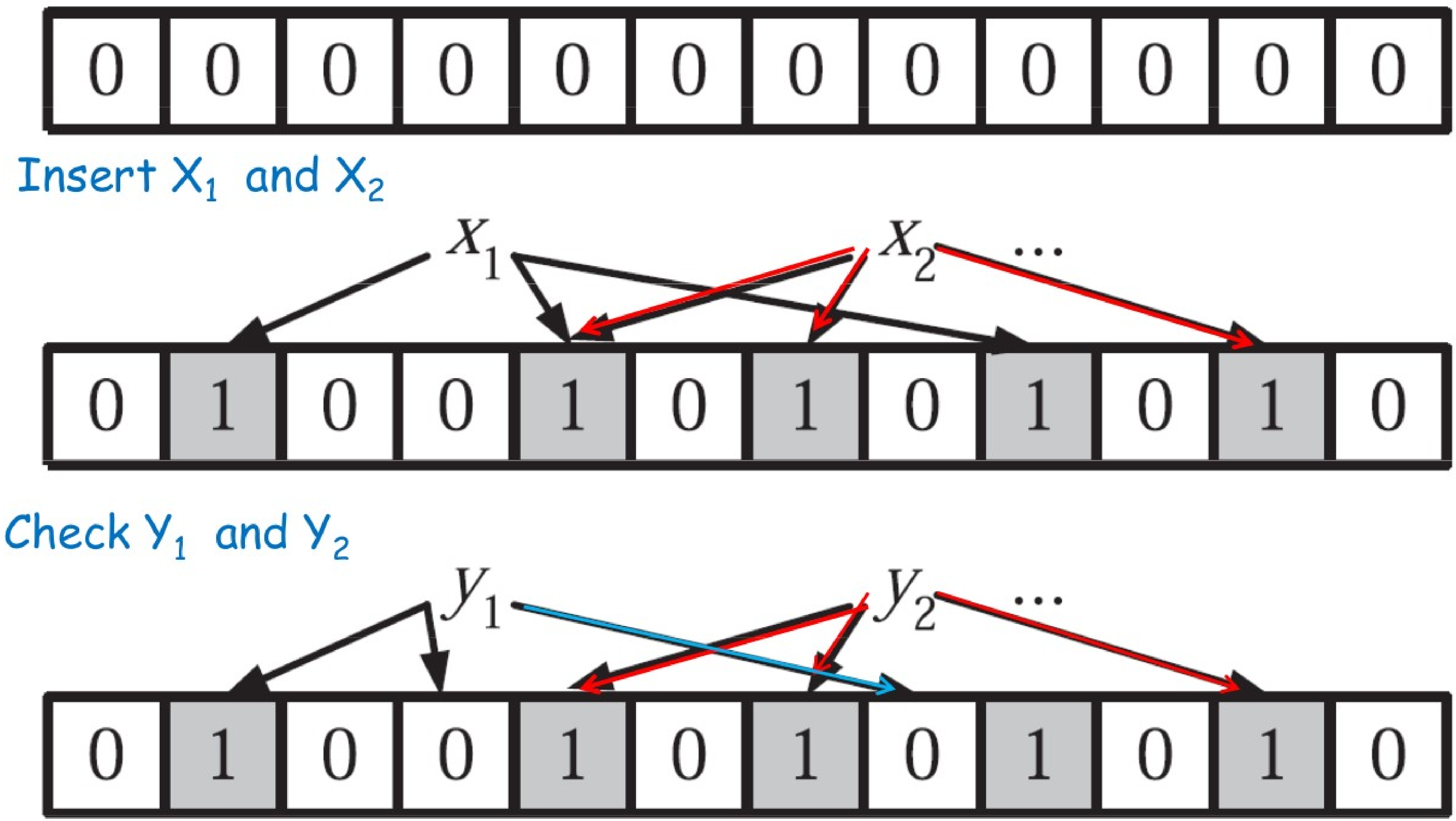
- Randomized data structure introduced by Burton Bloom [CACM 1970]
  - It represents a set for membership queries, with false positives
  - Probability of false positive can be controlled by design parameters
  - When space efficiency is important, a Bloom filter may be used if the effect of false positives can be mitigated.

- First applications in dictionaries and databases

Standard Bloom Filter

- A Bloom filter is an *array of m bits* representing a set $S = \{ x_1, x_2, \dots , x_n\}$ of *n* elements
  - Array set to 0 initially
- *k* independent hash functions $h_1, \dots , h_k$ with range $\{1, 2, \dots, m\}$
  - Assume that each hash function maps each item in the universe to a random number *uniformly* over the range $\{1, 2, \dots, m\}$
- For each element x in S, the bit $h_i(x)$ in the array is set to 1, for $1 \le i \le k$,
  - A bit in the array may be set to 1 multiple times for different elements

A Bloom filter example
(three hash functions)



Standard Bloom Filter (cont.)

- To check membership of y in S, check whether $h_i(y)$, $1 \le i \le k$, are all set to 1
  - If not, y is definitely not in S
  - Else, we conclude that y is in S, but sometimes this conclusion is wrong (false positive)
- For many applications, false positives are acceptable as long as the probability of a false positive is small enough
- We will assume that $kn < m$

False positive probability

- After all members of S have been hashed to a Bloom filter, the probability that a specific bit is still 0 is

$$p' = (1 - \frac{1}{m})^{kn} \simeq e^{-kn/m} = p$$

- For a non member, it may be found to be a member of S (all of its k bits are nonzero) with *false positive probability*

$$(1 - p')^k \simeq (1 - p)^k$$

False positive probability (cont.)

- Define

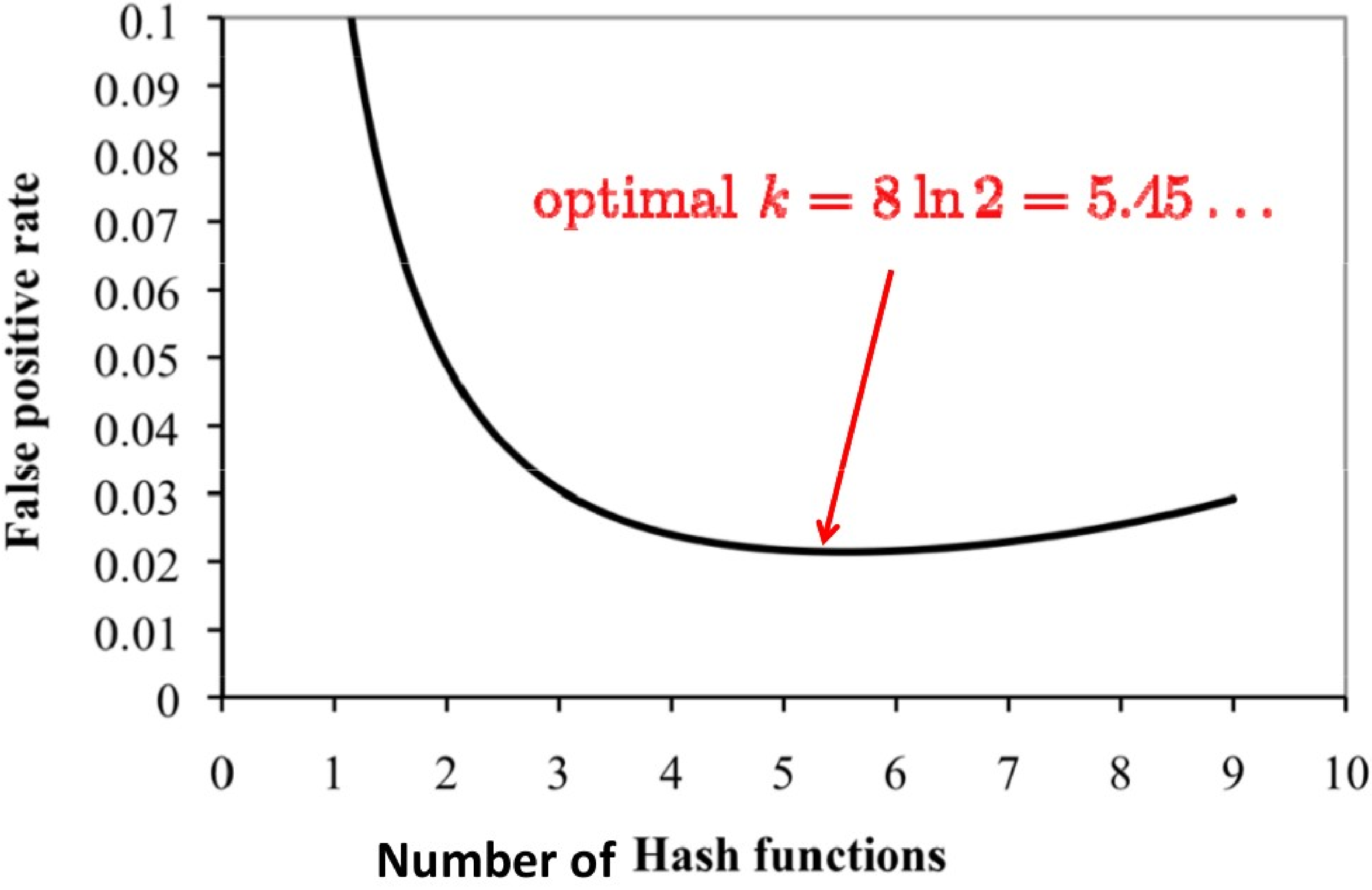
$$f' = (1-p')^k = (1-(1-\frac{1}{m})^{kn})^k$$

$$f = (1-p)^k = (1-e^{-kn/m})^k$$

- Two competing forces as k increases
  - Larger k -> $(1-p')^k$ is smaller for a fixed p'
  - Larger k -> p'= $(1-1/m)^{kn}$ is smaller -> 1-p' larger

False positive rate vs. k

Number of bits per member $\frac{m}{n} = 8$



Optimal number k from derivative

Rewrite f as

$$f = \exp(\ln(1-e^{-kn/m})^k) = \exp(k\ln(1-e^{-kn/m}))$$

Let $\boxed{g = k\ln(1-e^{-kn/m})}$

Minimizing g will minimize $f = \exp(g)$

$$\frac{\partial g}{\partial k} = \ln(1-e^{-kn/m}) + \frac{k}{1-e^{-kn/m}} \frac{\partial(1-e^{-kn/m})}{\partial k}$$

$$= \ln(1-e^{-kn/m}) + \frac{k}{1-e^{-kn/m}} \frac{n}{m} e^{-kn/m} = -\ln(2) + \ln(2) = 0$$

if we plug $\boxed{k = (m/n)\ln 2}$ which is optimal

(It is in fact a global optimum)

Optimal number of hash functions

- Using $k_{opt} = \frac{m}{n}\ln(2)$ the false positive rate is

$$(1-p)^{\frac{m}{n}\ln(2)} = (0.5)^{\frac{m}{n}\ln(2)} \simeq (0.6185)^{m/n}, \text{ where } \ln(2) = 0.6931$$

- In practice, k should be an integer. May choose an integer value smaller than k_{opt} to reduce hashing overhead

m/n denotes bits per entry

False positive rate

$$m/n = 6 \quad k = 4 \quad p_{\text{error}} = 0.0561$$

$$m/n = 8 \quad k = 6 \quad p_{\text{error}} = 0.0215$$

$$m/n = 12 \quad k = 8 \quad p_{\text{error}} = 0.00314$$

$$m/n = 16 \quad k = 11 \quad p_{\text{error}} = 0.000458$$

Counting Bloom filters

- Proposed by Fan et al. [2000] for distributed caching
- Every entry in a counting Bloom filter is a small counter (rather than a single bit).
  - When an item is inserted into the set, the corresponding counters are each incremented by 1
  - When an item is deleted from the set, the corresponding counters are each decremented by 1
- To avoid counter overflow, its size must be sufficiently large. It was found that 4 bits per counter are enough.