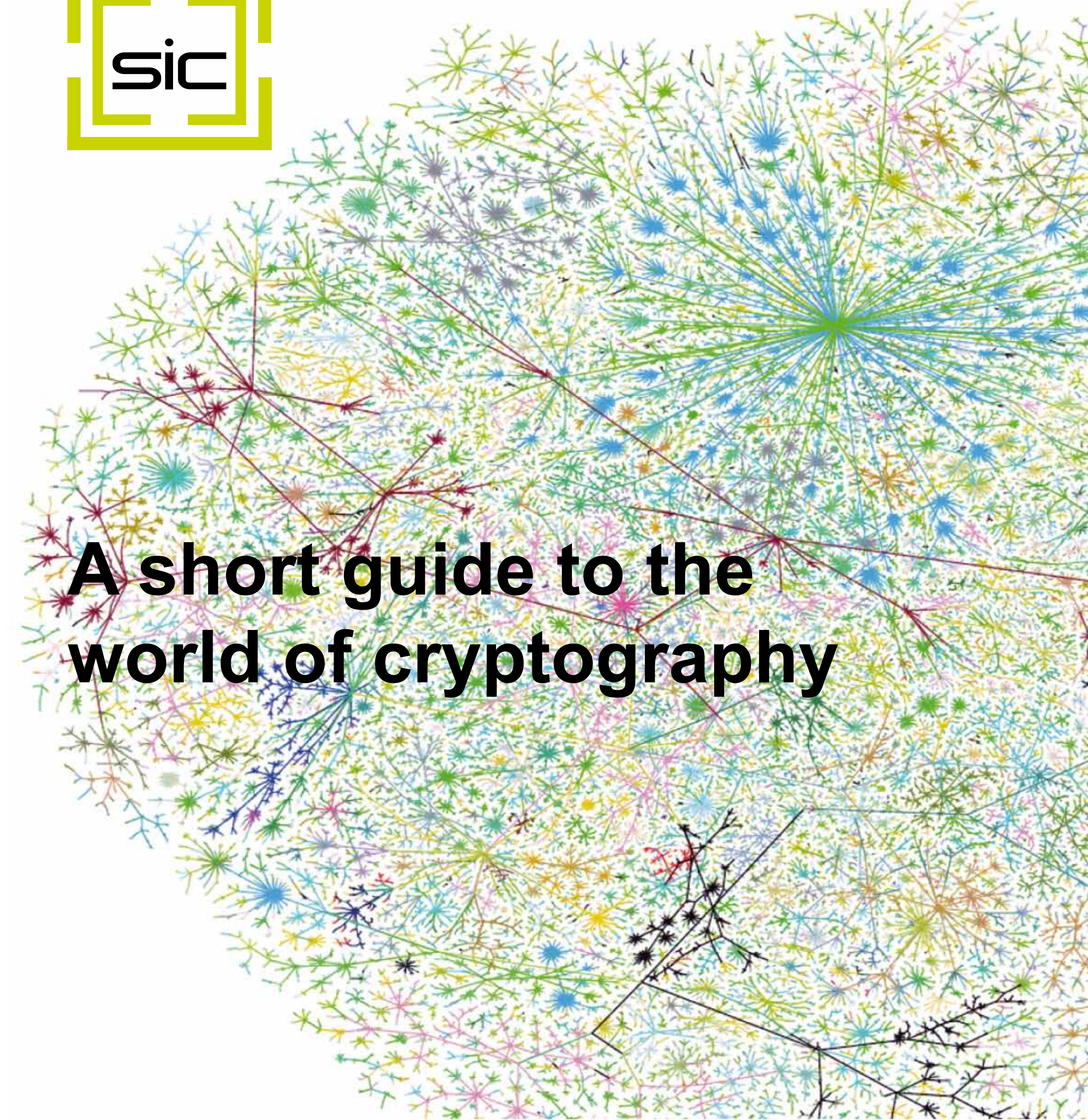
stiftung secure information and communication technologies

IAIK TUG



A short guide to the world of cryptography

INDEX

This brochure and its contents are published by:

SIC
Stiftung Secure Information and Communication Technologies
Inffeldgasse 16a
A-8010 Graz
Austria
Phone: +43 316 873 5582
Fax: +43 316 873 5593
https://jce.iaik.tugraz.at
Copyright (C) 2004 by Stiftung SIC

Part 1 - About SIC/IAIK

"Fantasy is hardly an escape from reality.
It's a way of understanding it." (Lloyd Alexander)

Chaper 1: Introduction

The Internet today poses many challenges to you as a software developer. You are forced to counter attacks you can't even envision or understand. You need to guarantee secure solutions to your customers. You want them to trust your software is immune against such attacks. How can you possibly achieve this?
Welcome to the world of cryptography. While cryptography will not provide a solution against all possible attacks, it is part of many solutions provided.
Cryptography forms the core of standards like TLS, used e.g. for secure communication in the Web, or like S/MIME, providing confidentiality and authenticity for emails. Cryptography is complex, and software developers who want to apply cryptographic functions or services in their applications need to understand at least some basics about it.
This brochure addresses newbies as well as highly skilled developers. It intends to provide newbies with just that basic information to help you getting started. Our experienced readership is invited to read about cryptography seen from a different and fairy-tale like point of view.

Who are we

Stiftung Secure Information and Communication Technologies SIC is a foundation that has been incorporated by the Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, and has been declared admissible in February 2003.
SIC's focus is on doing and sponsoring scientific research and development as well as teaching and knowledge transfer in the areas of applied information processing communications and information security. IAIK has transferred all rights of the IAIK-Java-crypto-toolkits to SIC on December 15, 2003. Since then, SIC is responsible for continuing development, support and sales of these toolkits.
SIC and IAIK continue to work together on the advancement of the toolkits. SIC will also sponsor research at IAIK, the result of which may be incorporated into the SIC product suite.

Our Products

SIC offers a wide range of cryptography- and securityrelated Java products, covering all aspects of data and communication security. This brochure will give you some basic ideas of services our toolkits can offer and will introduce their features to you.

Our Support

Additionally, we provide sophisticated support to our customers. Our support engineers who have worked in this field for many years are also authors of the supported software. We can therefore guarantee the best support possible.

Our Internationality

Additionally, we provideSIC and IAIK also actively participate in international standardisation activities, like ETSI-ESI, OASIS-DSS or relevant IETF and W3C-working groups. SIC and IAIK are also partners in international projects, like the FP6-projects OPEN-TC, POSITIF, ECRYPT, USB-CRYPT or SCARD. These international contacts guarantee the high level of knowledge available and influence the products and service we provide. sophisticated support to our customers. Our support engineers who have worked in this field for many years are also authors of the supported software. We can therefore guarantee the best support possible.

Our Commitment

Excellent security products require excellent quality. SIC and IAIK strive to attain that quality throughout our product suite. One of the various proofs of quality is the independent evaluation of our software. TÜViT-Essen (Germany 2004, EAL3+/hoch) and JISEC (Japan 2007, EAL3, internationally accepted) have certified the IAIK-JCE toolkit according to Common Criteria. The fact that we have been on the market since 1997, guarantees continuity and ensures quality - many of our customers have had access to the source codes to ensure themselves of the quality therein.

Chap. 2: Basics of Cryptography

Here Legrand, having re-heated the parchment, submitted It my inspection. The following characters were rudely traced, in a red tint, between the death's-head and the goat:

53++!305))6*;4826)4+.)4+);806*;48!8`60))85;]8*
:+*8!83(88)5*!;46(;88*96*?;8)*+(;485);5*!2:*+(
;4956*2(5*-4)8`8*;4069285);)6!8)4++;1(+9;4808
1;8:8+1;48!85;4)485!528806*81(+9;48;(88;4(+?3
4;48)4+;161;:188;+?;

"But," said I, returning him the slip, "I am as much in the dark as ever. Were all the jewels of Golconda awaiting me on my solution of this enigma, I am quite sure that I should be unable to earn them."

"And yet," said Legrand, "the solution is by no means so difficult as you might be led to imagine from the first hasty inspection of the characters. These characters, as any one might readily guess, form a cipher --that is to say, they convey a meaning; but then, from what is known of Kidd, I could not suppose him capable of constructing any of the more abstruse cryptographs. I made up my mind, at once, that this was of a simple species --such, however, as would appear, to the crude intellect of the sailor, absolutely insoluble without the key."

"And you really solved it?"

Edgar Alan Poe, The Gold-Bug

Yes, he solved this problem. If Captain Kidd had known more about cryptography, Legrand would not have solved this riddle like that easily. The following paragraphs will tell you some basics of cryptography.
We will show how to protect two different things: messages and connections. A message can be any block of data with a limited length. In practice, this is usually a single file, an email message, a data block in memory, or an object in a database. Stream connections are reliable transport media like TCP network connections or pipes. In contrast to messages, these connections can transport data of arbitrary length. In addition, the receiver can already process parts of the data when the sender has not yet sent all data or does not even know the remaining length.
What type of protection can we offer?

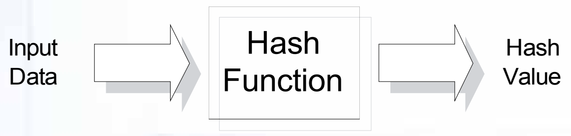
Well, with cryptography we can ensure the integrity, the authenticity, and the confidentiality of data. Integrity means that the receiver of data can detect any modification of the data on the way from the sender to the receiver. Authenticity is a property that allows verifying that the data really originates from the alleged sender. With confidentiality we mean a feature that enables the sender to transform the original data in a way that only the designated recipient can reconstruct it. No eavesdropper between sender and receiver can recover the original data.
There are different types of cryptographic operations which offer different kinds of protection. We will have a look at the cryptographic operations called hash algorithms, signature schemes, and ciphers.

Hash Functions

One of the most important cryptographic operations are hash functions. A hash function is an operation which takes data of arbitrary length as input and produces an small output of a fixed size (e.g. 160 bit). This small hash value serves as a representative for the original data. For instance, signatures use this hash value instead of the original data. The hash value is a kind of fingerprint of the complete data.
To be secure, a hash function has to meet certain requirements.

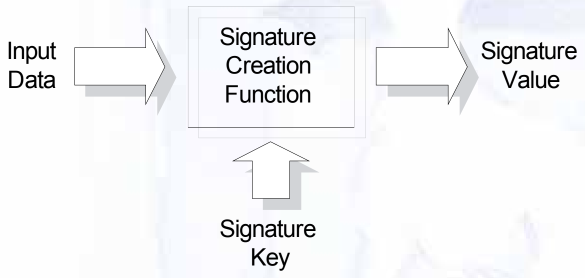
Given a hash value, it must be infeasible to compute any input data for which the hash functions produces this value.
It must be practically impossible to create two different input data for which the hash function produces the same hash value.

The hash value can be used to ensure integrity of data. If an application knows the hash value of some original data, it can easily verify the integrity of received data. The application simply calculates the hash value over the received data and compares this value with the original value. If both match, the data is complete and unmodified.

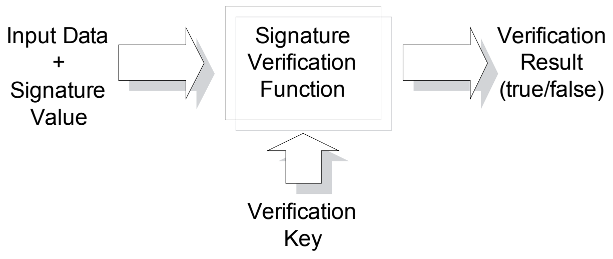


Signature Schemes

In addition to integrity, a signature scheme can ensure authenticity of data. A signature scheme consists of two functions: a signature creation function and a signature verification function.



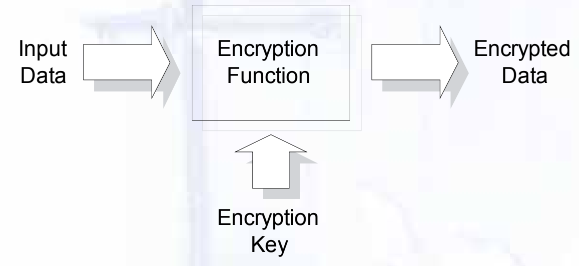
The signature creation function takes data as input and a signature key as parameter. The resulting signature value is of a fixed size (e.g. 1024 bit), similar to a hash value.


Input Data + Signature Value
Signature Verification Function
Verification Result (true/false)
Verification Key

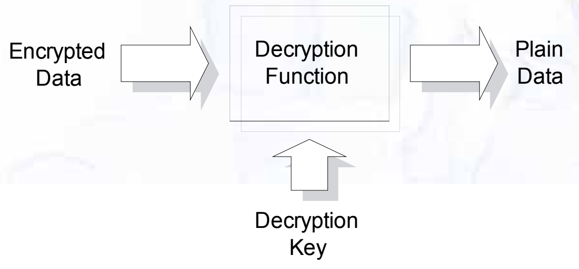
The signature verification function takes the data and the signature value as input, and the verification key as parameter. The result is true or false.
The verification key is a key corresponding to the signature key which has been used to calculate the given signature value. The result is true if the data is complete and unmodified, and the signature value has been calculated with the signature key corresponding to the used verification key. Otherwise the result is false, and the signature must be rejected.
In a secure signature scheme it is impossible to create a valid signature without the signature key.
Usually, the signer generates a key pair, keeps the signature key secrete and publishes the verification key along with his name. A relying party checks if the verification key belongs to the alleged signer. If a signature verifies with a verification key which is bound to an entity, this evidences that this entity created the signature. With technical and organizational means it is possible to ensure that only the legitimate owner can use the signature key.
Moreover, using advanced technologies to get authentic verification keys bound to uniquely identifiable entities, cryptographic signatures can serve as an electronic substitute for handwritten signatures. In combination with such techniques, a signature can provide a property called non-repudiation. This means that a verifiable signature can be considered as evidence that a certain person created the signature value, and this person cannot deny this fact.

Ciphers

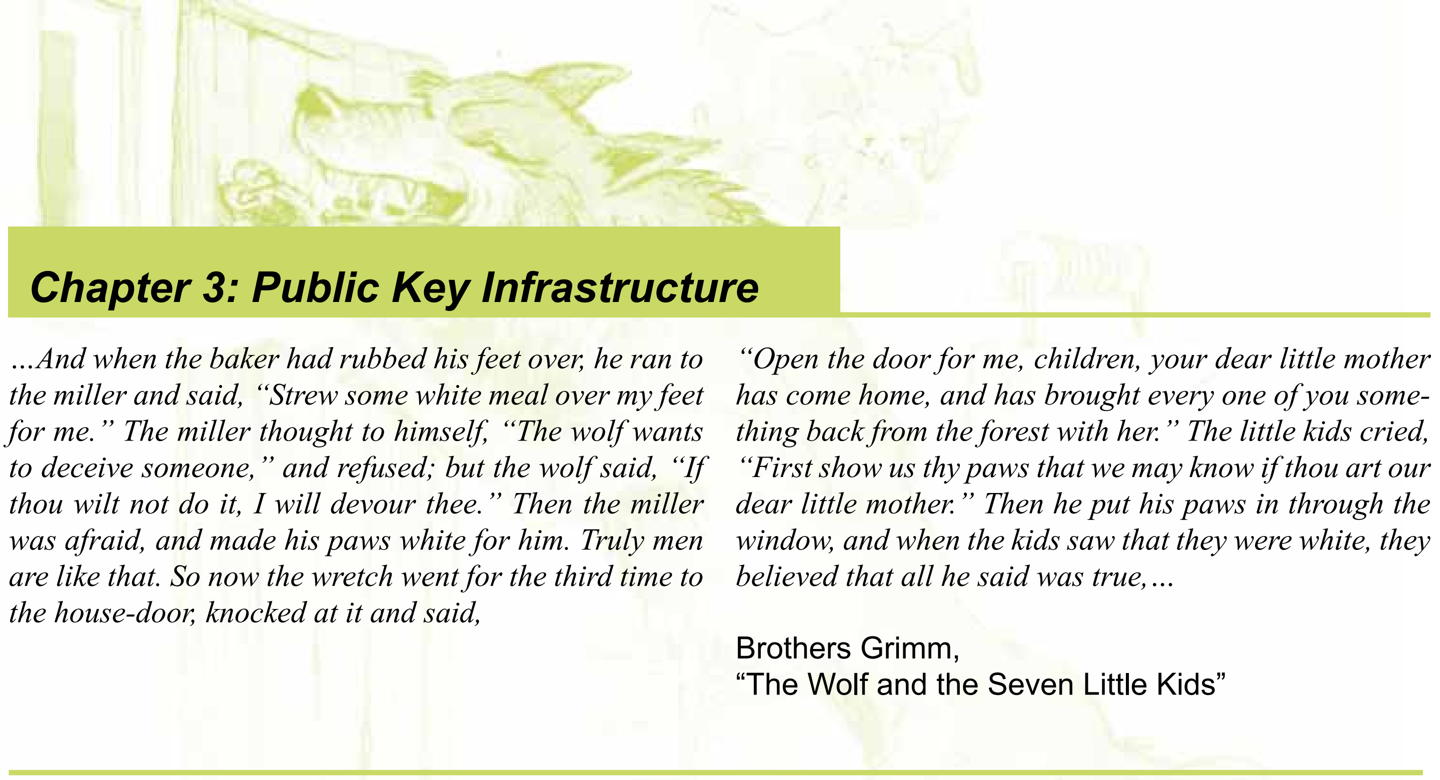
To ensure confidentiality we can employ a cipher. There is one function for encryption and another for decryption.



The encryption function takes the confidential data as input and an encryption key as parameter. As output we get the encrypted data. It is safe to send this encrypted data via insecure media like the internet.



For decryption, we have another function. It requires the encrypted data as input and the decryption key as parameter. If the decryption key corresponds to the key used for encryption, we get the original data as result. Provided that used cipher is secure, there is no way to decrypt the data without this decryption key. The decryption key must be kept secret because it is everything required to recover the confidential data.
We can distinguish two types of ciphers. One type uses the same key for encryption and decryption; the sender and receiver must share the secret key. This type is called symmetric cipher. In contrast, asymmetric ciphers use key pairs, consisting of an encryption key and a decryption key. The advantage is that one must only keep the decryption key secret. The encryption key can be published. The drawback is that asymmetric ciphers are about 1000 slower than symmetric ciphers.



Chapter 3: Public Key Infrastructure

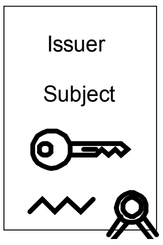
…And when the baker had rubbed his feet over, he ran to the miller and said, "Strew some white meal over my feet for me." The miller thought to himself, "The wolf wants to deceive someone," and refused; but the wolf said, "If thou wilt not do it, I will devour thee." Then the miller was afraid, and made his paws white for him. Truly men are like that. So now the wretch went for the third time to the house-door, knocked at it and said, "Open the door for me, children, your dear little mother has come home, and has brought every one of you something back from the forest with her." The little kids cried, "First show us thy paws that we may know if thou art our dear little mother." Then he put his paws in through the window, and when the kids saw that they were white, they believed that all he said was true,…

Brothers Grimm,
"The Wolf and the Seven Little Kids"

… but the youngest little goat was still suspicious and cried: "Show us your certificate first, so that we are sure that you are our dear mother!" So the wolf had to go away again, and since he did not know what a certificate was, he asked and learned that he could get one from a certification authority. But no cunning and trickery, and no thread did help him: nobody believed him that he was the goat with the seven little kids and nobody was willing to give him a certificate. Just when he decided to give up he met another wolf who told him about a certification authority that would not ask whether he was a wolf or a goat. The wolf jumped for joy, and when the day was over he had a certificate saying that he was the goat with the seven little kids. So he went to the goat's house for the fourth time, knocked and said, "Open the door for me, children, your dear little mother has come home, and has brought every one of you something from the forest with her." And he put his white dyed paws and his certificate in through the window. The kids discussed about it and then the youngest kid cried: "Sorry, we cannot open the door. You are not our mother. We are not allowed to trust the authority that has issued your certificate!" Hearing this, the wolf got so furious that he ate up his useless certificate.
The certificate, however, stuck in his throat so that he could not breathe anymore and finally dropped dead. Now the seven little kids opened the door, danced around the dead wolf and sang: "The wolf is dead! The wolf is dead!"

A Matter of Trust

Not trusting the certificate of the wolf has saved the lives of the seven little goats. However, what does this mean for the world of E security? Why is trust so important? For example, when receiving a signed message, shouldn't it be sufficient to verify its signature? The answer is: No! Verifying the correctness of the signature will only tell us that the message has not been tampered with and that it has been signed with the signer's private key. It would not tell us who the signer actually is – or, more precisely, if the signer really is who she/he pretends to be. For verifying the signature we only need the signer's public key; for verifying her/his identity we need her/his certificate.



A certificate binds the public key of some person to her/his name. Certificates are issued by Certification Authorities (CAs) and are usually made publicly available via HTTP download or LDAP databases. Besides name and public key of its owner (subject), a certificate also contains the name of the issuing CA, a unique serial number, and (like a passport) an issuing and expiration date. All these items – and any number of certificate extensions with additional information like, for instance, certificate purpose or email address of the owner – are put together and signed with the private key of the CA.

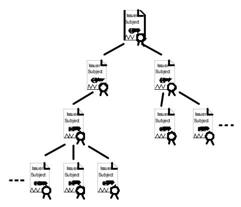
With this signature the CA confirms that the public key included in the certificate belongs to the entity that is the owner of the certificate. Since the CA is responsible for the information given in a certificate, it should carefully check the identity of the requesting individual. Carefully? Doesn't the certificate of the wolf say that he is the goat with the seven little kids? Of course, this is only a fairy tale, but in any fairy tale there may be a little bit of truth. And in real life there are CAs that may issue certificates to anybody who comes along. Such certificates, however, may only be used for testing or demonstration purposes. The more critical the purpose of a certificate is, the more carefully a CA's registration authority has to investigate the identity of the requesting entity. Especially authorities that issue Qualified Certificates for use with advanced electronic signatures in the scope of E commerce and E government must take care for a high assurance level. In any case, before putting trust into a certificate, we should read the policy or practice statement of a CA. If we then do not agree with the CA's way of working we will not trust the certificates it issues.
Hopefully we have learned our lesson from the seven little kids!

Passports May Be Revoked

If a certificate is suspected to get misused (e.g. because the corresponding private key has been compromised), it has to be revoked before its validity period has expired. Revocation means that the CA puts the certificate's serial number on a publicly available revocation list (CRL). If we are verifying a certificate, we don't have to forget to look at the CRL to ensure that the certificate is still valid. Since CRLs may grow with their entries, more and more CAs provide revocation information via the more efficient client-server based Online Certificate Status Protocol (OCSP).

We Build a Certification Tree

Of course, not only human beings may get a certificate; a CA also may certify organizations, web-servers, etc. and other (sub) CAs. Each sub CA itself again may certify any number of further CAs and so a tree-like certification structure may pop up having final end user certificates at it leaves and one single top level CA certificate at its root. Since nobody can reside above the root of the tree, the top level CA holds a self-signed certificate; i.e. it certifies its public key with its own private key. Building and validating a path that leads from some leaf of the tree to the root is often a very tricky action. However, usually we do not have to walk the whole way; we can stop as soon as we have found a CA in the path we explicitly want to trust. Such a CA is called a trust anchor.



Again: it's a Matter of Trust

All together, certificates, revocation lists, certification authorities, certified entities, etc. are what we call a Public Key Infrastructure (PKI). Unlike the famous web of trust of the popular PGP protocol, the X.509 PKI system of the PKIX working group at IETF has propagated a more hierarchical trust model that is based on certification authorities, as described above. This makes X.509 more suitable for the large Internet community: if we place trust in one particular CA, we also trust any CA that is located beyond this trust anchor in the tree; down to the end users at the final leaves. The seven little kids did not trust the CA that has issued the certificate of the wolf, and consequently they did not trust the wolf himself. So finally we have returned to the point where we have started our journey: it is all a matter of trust!

Chap. 4: Key Management

...Ali Baba saw the robbers, as soon as they came under the tree, each unbridle his horse and hobble it. Then all took off their saddlebags, which proved to be full of gold and silver. The man who seemed to be the captain presently pushed forward, load on shoulder, through thorns and thickets, till he came up to a certain spot, where he uttered these strange words: "Open, Sesame!" And forthwith appeared a wide doorway in the face of the rock. The robbers went in, and last of all their chief, and then the portal shut of itself.

Long while they stayed within the cave whilst Ali Baba was constrained to abide perched upon the tree, reflecting that if he came down, peradventure the band might issue forth that very moment and seize him and slay him. At last he had determined to mount one of the horses and driving on his asses, to return townward, when suddenly the portal flew open. The robber chief was first to issue forth, then, standing at the entrance, he saw and counted his men as they came out, and lastly he spoke the magical words, "Shut, Sesame!" whereat the door closed of itself. When all had passed muster and review, each slung on his saddlebags and bridled his own horse, and as soon as ready they rode off, led by the leader, in the direction whence they came. Ali Baba remained still perched on the tree and watched their departure, nor would he descend until what time they were clean gone out of sight, lest perchance one of them return and look around and descry him.

Then he thought within himself: "I too will try the virtue of those magical words and see if at my bidding the door will open and close." So he called out aloud, "Open, Sesame!" And no sooner had he spoken than straightway the portal flew open and he entered within....

"Ali Baba and the forty thieves"

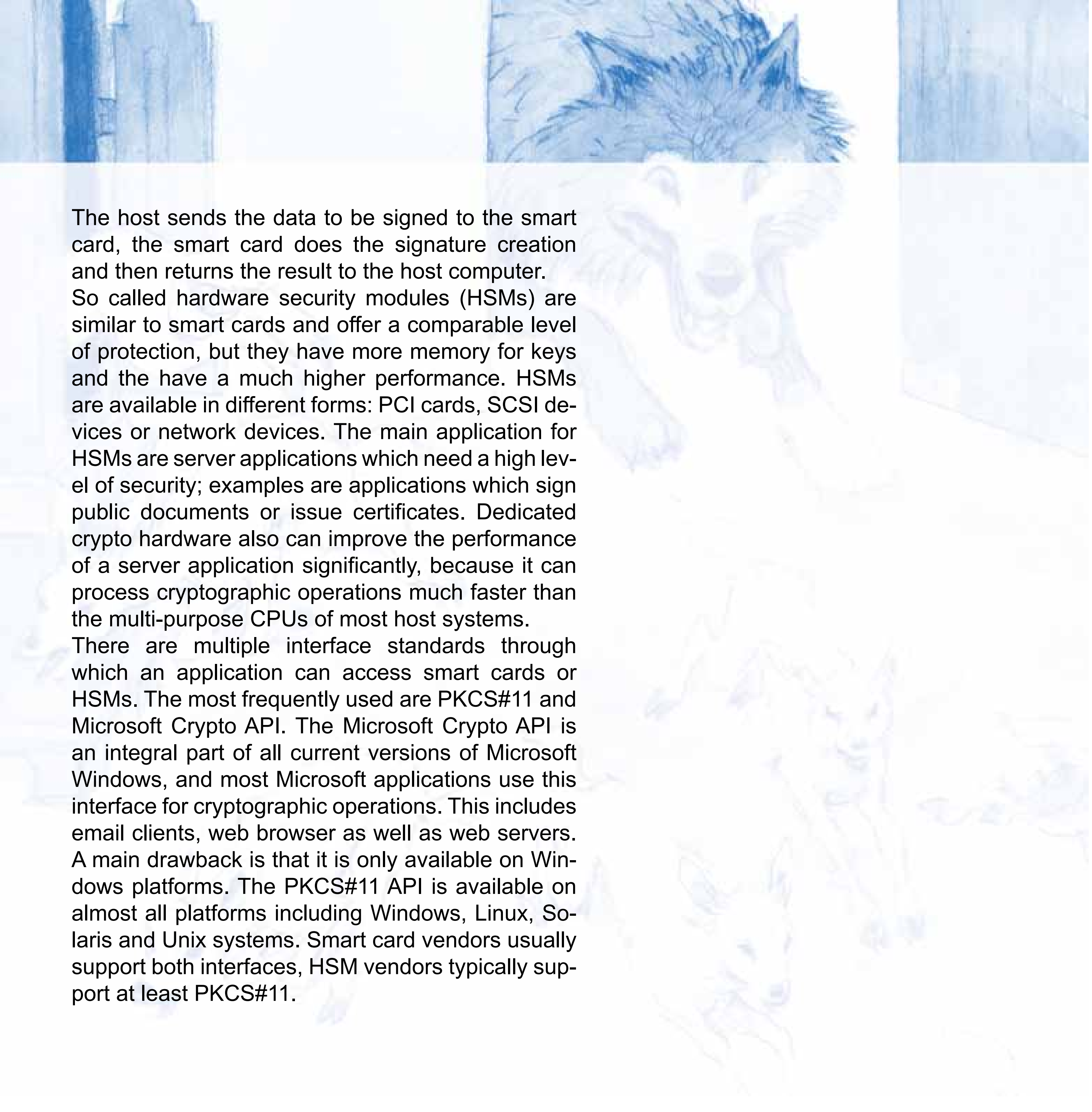
The words "Open, Sesame!" were the key to the cave of the thieves. By using it carelessly, the thieves enabled Ali Baba to hear these words easily.

For all cryptographic operations which use keys, it is essential to handle the keys appropriately. For instance, if an application uses a cipher to protect data, it must protect the decryption key. A system cannot be safer than its weakest part. An insufficiently protected sensitive key can compromise the complete system. In the case of digital signatures, key protection can be especially critical. If signatures are expected to offer non repudiation, a signer cannot deny having created a signature if it verifies with a certain verification key. Therefore, an attacker who is able to steal a signature key can create valid signatures in the name of the key owner, and the owner may be liable for these forged signatures. An adequate key protection is indispensable.

There are several methods commonly used in practice. They offer a different degree of protection; this means, some of them can withstand attacks with a higher attack potential than others. The level of protection needed depends on the value of the assets to be protected. Private emails with no information of public interest may be a less attractive target than sensitive information of a government.

The simplest method to keep a key is to store it in plain on a disk or some other persistent memory. This method is only as safe as the disk. If someone gains access to the disk, it is easy to make a copy of the key. In practice, keys are rarely stored in plain. Usually, they are protected with a password or PIN. This requires having access to the storage media and knowing the password.

For higher security, there are more advanced technologies. Dedicated hardware like smart cards can protect against heavy attacks. Even security experts with a big budget are unable to extract sensitive keys from a modern smart card. The essence of cryptographic smart cards is that they are constructed in a manner that the keys can never leave the card. The cards have a processor and software, which generate the keys, use the keys and destroy the keys. The cards do not have any interface to extract the keys—for nobody. Thus, the cards work differently than a simple disk. They are not just a key storage, they are rather computers themselves. It is no longer the host system which does the actual cryptographic operation like signature creation.



The host sends the data to be signed to the smart card, the smart card does the signature creation and then returns the result to the host computer.
So called hardware security modules (HSMs) are similar to smart cards and offer a comparable level of protection, but they have more memory for keys and the have a much higher performance. HSMs are available in different forms: PCI cards, SCSI devices or network devices. The main application for HSMs are server applications which need a high level of security; examples are applications which sign public documents or issue certificates. Dedicated crypto hardware also can improve the performance of a server application significantly, because it can process cryptographic operations much faster than the multi-purpose CPUs of most host systems.
There are multiple interface standards through which an application can access smart cards or HSMs. The most frequently used are PKCS#11 and Microsoft Crypto API. The Microsoft Crypto API is an integral part of all current versions of Microsoft Windows, and most Microsoft applications use this interface for cryptographic operations. This includes email clients, web browser as well as web servers. A main drawback is that it is only available on Windows platforms. The PKCS#11 API is available on almost all platforms including Windows, Linux, Solaris and Unix systems. Smart card vendors usually support both interfaces, HSM vendors typically support at least PKCS#11.

Chap. 5: Secure Connections – SSL and TLS

“I am overhearing them.”
“Pshaw! Sneaking and overhearing seems to be your favorite pleasure! But keep in mind, that now you are neither at the boondocks nor in the dark and bloody grounds!”
“What does it matter? Shouldn’t it be possible to eavesdrop here, too?”
“Well; but you find out nothing from it.”
“You find out all what you want to know!”

Karl May, “An der Tigerbrücke” (“Am stillen Ozean”)

But just in case the conversation is not protected by SSL!
Sure, Old Shatterhand is a well languaged guy and a real champion when sneaking and overhearing hostile Indians. But none of his abilities would help him when trying to crack the AES or TripleDES algorithm in order to find out where the Internet Tramps will complete their next coup. Poor Charlie, you would be lost in the era of E-security!

However, if data is exchanged via the Internet without any cryptographic protection, today it would be much easier to steal the required information. Cautious and dangerous sneaking actions wouldn’t be necessary anymore! For this reason you should be aware of the risks you may take when exchanging critical data over the Internet. And it would probably not be a well-behaved fellow man like Winnetou or Old Shatterhand who is overhearing you for taking advantage of unauthorizedly accessed information.

Do Not Send Your Data Via Plain HTTP!

Just imagine a situation where you are ready to finish your order in an online store by entering your credit card number. Before pressing the ok button to send your confidential data to the store – and possibly to any eavesdropper that is sniffing on the line – you should make sure that the data will not travel over plain HTTP only. Neither TCP nor HTTP provides any kind of protection mechanisms against passive (where the “man in the middle” can read your data) or active (where he is going to modify it) attacks. This might be ok if you are surfing the Web for fun during your spare time, but is not acceptable for critical applications like online shopping, telebanking or E-government.

HTTPS: The “S” Makes It Secure!

SSL turns HTTP into HTTPS, and from now on we are on the secure side! SSL stands for Secure Sockets Layer. It is an application-independent client-server security protocol that operates over a reliable transport protocol (typically TCP). With the help of SSL, any higher level application protocol like HTTP, FTP or TELNET, can be protected by means of the following security services:

- Server- (and maybe client-) authentication using public key cryptography
- Confidentiality of the data by using symmetric cryptography
- Integrity of the data by using message authentication codes

The SSL protocol has been developed by Netscape. The original version (SSL 2.0) had some serious limitations and security problems, and it has therefore been replaced by its thoroughly re-designed successor version SSL3.0. TLS 1.0 (Transport Layer Security) has been standardized by the Internet Engineering Task Force (IETF). It is nearly identical to SSL3.0, containing some minor updates. In the course of time, several attacks have been published against the SSL/TLS protocol, some of them less, some of them more serious ones, but none of them launched by Old Shatterhand! Effective countermeasures against all these attacks have been developed and implemented.

It Starts With a Handshake and Ends With a Secure Channel

TLS security is channel-based. First client and server go into an initial handshake phase to agree upon a so called cipher suite, defining the cryptographic parameters for the following session. As soon as the handshake is finished, a secure channel is set up. From now on any data that travels over the channel is protected by applying the cryptographic mechanisms negotiated during the handshake. The general handshake proceeding may look like the following conversation: The client starts the handshake and invites the server to install a secure connection: The client starts the handshake and invites the server to install a secure connection: “Hello, Mr. Server. How are you? My boss has told me to set up a secure conversation. Here are my favorite cipher suites: First I would kindly ask you to authenticate yourself with an RSA certificate. I am sorry, but this is the only type of certificate I am able to read! However, I am more flexible with respect to the symmetric cipher algorithm. Although I would prefer to use AES for encrypting the data, it would also be ok to take RC4 or TripleDES. The MAC calculation should be done by means of the SHA 1 algorithm! Please tell me if any of these options would be suitable for you!”

Since Mr. Server is able to support all the favorite algorithms of the client, he sends his approval: “Have a nice day, Mr. Client! RSA, AES and SHA 1 are ok for me. Here comes my certificate. I am very sorry, but I have to ask you to authenticate yourself, too. Please send me your certificate!”

The client, after verifying the server’s certificate, now in response has to send his own certificate to fulfill the client authentication request. If he does not do this (maybe he is not able to) it is up to the server to say “Sorry, but I cannot let you in without your certificate! Farewell and Goodbye”, or he can accept the anonymity of the client.

After creating and (securely) exchanging a secret session key, client and sever start the secure data exchange: “Now we say goodbye to all the spies on the line. Let the encrypted data flow!”

And Mr. Shatterhand? He would probably say something like: “Hell and damnation! What’s that? I hear the voice, but I can’t understand anything!”

Chapter 6: Secure Messaging

After a year the King had to take the field, so he commended his young Queen to the care of his mother and said, "If she is brought to bed take care of her, n u r s e her well, and tell me of it at once in a letter." Then she gave birth to a fine boy. So the old mother made haste to write and announce the joyful news to him. But the messenger rested by a brook on the way, and as he was fatigued by the great distance, he fell asleep. Then came the Devil, who was always seeking to injure the good Queen, and exchanged the letter for another, in which was written that the Queen had brought a monster into the world. When the King read the letter he was shocked and much troubled, but he wrote in answer that they were to take great care of the Queen and nurse her well until his arrival. The messenger went back with the letter, but rested at the same place and again fell asleep. Then came the Devil once more, and put a different letter in his pocket, in which it was written that they were to put the Queen and her child to death. The old mother was terribly shocked when she received the letter, and could not believe it. She wrote back again to the King, but received no other answer, because each time the Devil substituted a false letter, and in the last letter it was also written that she was to preserve the Queen's tongue and eyes as a token that she had obeyed.

Brothers Grimm, "The Girl without Hands"

Diabolic competition for Bruce Schneier's famous crypto villains Eve and Mallory? Nobody else than Lucifer himself has left the hell house to come on earth and tamper the correspondence of honest mortals! He does not only ignore the privacy of letters and reads a message which was not addressed to him, he also makes an active attack and forges the contents of any letter he can catch up. We do not really believe that the story would take a better course if we would transfer it to the present day where king and mother would have the chance to benefit from the achievement of electronic mail. We are afraid that they again would be innocent enough to send their messages in clear over the internet. An attacker wouldn't even have to unseal a nonexistent cover for reading the contents of a message and forge it as he like. Sure, we cannot be aware about the cryptographic genius of the Devil; nevertheless we want to show you how you can protect your electronic documents and messages against attacks that are not supernatural.

From SMTP to MIME; from MIME to S/MIME

Internet mail is based on the Simple Mail Transfer Protocol (SMTP). SMTP, however, restricts messages to contain only 7 bit US-ASCII characters. This may be suitable for pure text messages, but not for multimedia data like pictures, audio or video material. MIME (Multipurpose Internet Mail Extensions) extends SMTP about the ability to handle arbitrary binary data. Both, STMP and MIME do not contain any security mechanisms for protecting a message when traveling through the wide and open internet. This drawback has finally led to the introduction of the S/MIME (Secure MIME) protocol. Today any notable email client, like Microsoft Outlook or Mozilla Messenger, is able to speak S/MIME

S/MIME relies on CMS

S/MIMEv2 gets its cryptographic capabilities from the PKCS#7 protocol, which specifies several cryptographic mechanisms for digitally signing, digesting, encrypting or authenticating any kind of data. The successor versions of PKCS#7 and S/MIMEv2, CMS (Cryptographic Message Standard) and S/MIMEv3, respectively, take both care of algorithm independence and they introduce additional content formats and processing rules. Furthermore, S/MIMEv3 has been enhanced by new security features especially designed for business and finance applications.

Digital Signature and Digital Envelope

Based on the cryptographic functionalities of CMS, S/MIME provides the following security services:

- Authenticity, Integrity and Non-repudiation by means of digital signatures
- Privacy and Confidentiality by means of digital envelopes

When signing a document with your private key you make sure that it cannot be forged. At the same time you agree on being responsible for the content of the document. For providing proof of your identity, you add your certificate to the document.

A digital envelope is a hybrid technique. It uses a combination of asymmetric and symmetric cryptography to encrypt a document for some indented recipient only. Since asymmetric cryptography is comparably slow, you first create a symmetric (temporary) content encryption key and use it to encrypt the content data. Then you encrypt the content encryption key with the recipient's public key retrieved from her/his certificate – and send both, encrypted content and encrypted secret key to the recipient. Now the recipient uses her/his private key to decrypt the content encryption key which finally enables him to decrypt the encrypted content.

As you can see above, any cryptographic information like certificate or content encryption key that may be required by the recipient for verification or decryption, respectively, is packed with the message itself. This is an essential difference to connection-based protocols like TLS, where all the data is exchanged over a secure channel.

XMLDSIG and XML Encryption

Unlike CMS, which encloses signature and encryption formats into one single specification, the XML world defines two separate standards: XMLDSIG and XML ENCRYPTION.

The "XML Signature Syntax and Processing" standard (XMLDSIG) describes the format of a signed message in XML. It specifies the processing necessary to sign and verify such a signed message, respectively. Example applications are security features of markup languages such as Security Assertion Markup Language (SAML), or XML-based protocols like Simple Object Access Protocol (SOAP). The Austrian government makes heavy use of XMLDSIG for its E-Government services ("Austrian Citizen Card").

The W3C standard "XML Encryption Syntax and Processing" specifies the way how to encrypt, respectively decrypt, XML data as well as arbitrary data and represent the encryption information as XML. It is closely related to the XMLDSIG standard.

Conclusion

And the morale of the whole story? If king and mother would have signed their letters, the Devil would have not been able to forge their messages; if they additionally would have encrypted them, he would not have been able to read their contents…

PART 2 - Our Products
“Fairy Tale is not true, but there is a hint in it.”

The following chapter wants to give you a brief survey on the various products we offer to our customers. These products allow easy embedding of data security features in your applications, they are 100% pure Java, and they offer you high-performing and efficient data security functionality.

Our products offer various cryptographic algorithms including PKI solutions. Further highlights are easy integration of creation and verification of XML-based digital signatures into applications.

IAIK-JCE

The IAIK JCE toolkit contains a CSP provider for the JCA/JCE framework of the Java™ platform. Thus, it offers a wide range of cryptographic algorithms via the standard Java™ API for cryptographic operations. In addition to the CSP, the toolkit comes with a JCE implementation, which allows its use with any JDK version since 1.2. Despite the cryptographic algorithms, this toolkit supports several other protocols and message formats out-of-the-box.

Feature List

• Implemented entirely in the Java™ language

• Support for JDK Versions 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 and compatible

• Reliability and robustness through use in real-world applications over many years

• Developed and maintained by experts in applied cryptography and the Java™ programming language

• Comprehensive documentation and many demos

• Hash algorithms SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, MD2, MD5, RIPEMD-128, RIPEMD-160, RIPEMD-256, RIPEMD-320, WHIRLPOOL

• SHA-3 candidate hash algorithms: BLAKE-224, BLAKE-256, BLAKE-384, and BLAKE-512, Groestl-224, Groestl-256, Groestl-384, and Groestl-512, JH-224, JH-256, JH-384, and JH-512, KECCAK-224, KECCAK-256, KECCAK-384, and KECCAK-512, as well as Skein-224, Skein-256, Skein-384, and Skein-512

• Signature schemes
  - PKCS#1 version 1.5 RSA with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 , MD2, MD5, RIPEMD-128, RIPEMD-160, Whirlpool, GOST-3411;
  - PKCS#1 version 2.1 RSA PSS with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD2, MD5, RIPEMD-128, RIPEMD-160, RIPEMD-256, Whirlpool; Support for RSA PSS keys (RFC 4055)
  - ISO 9796-2 RSA with SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-128, RIPEMD-160, Whirlpool;
  - raw RSA with ext. hashing
  - SSL/TLS RSA signature with MD5, SHA-1
  - DSA and DSA with external hashing
  - DSA with SHA-2 (FIPS 186-3)

• Symmetric ciphers AES, DES, Triple-DES (112 and 168 bit), IDEA, Blowfish, Camellia, GOST, CAST-128, RC2, ARCFOUR (compatible with RC4™), RC5, RC6, password-based (PKCS#5 PBES1 with MD5, SHA-1 and DES, Triple-DES, RC2; PKCS#5 PBES2 with AES, DESede, ... and HMAC/SHA-1, HMAC/SHA-2), MARS, Twofish, Rijndael, Rijnael-256, Serpent
  - key generation
  - Password based key derivation (PKCS#5, PBKDF2; PKCS#12)
  - encryption and decryption
  - ECB, CBC, PCBC, CFB, OFB, CTR, CCM, GCM, CTS modes of operation
  - padding PKCS#5, SSL version 3.0, ISO 7816-4, ISO 10126-2, no padding

• Optional AES addon which makes use of the AES-NI instruction set extensions of modern x86 CPUs

• Key-Wrap ciphers for AES, Camellia, Triple-DES, CAST, RC2, HMAC-AES, HMAC-Triple-DES

• RSA encryption/decryption according to PKCS#1 version 1.5 and 2.1 OAEP; Support for RSAES-OAEP keys according to RFC 4055

• Blinding for RSA private key operations

• HMAC with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, MD5, RIPEMD-128, RIPEMD-160, Whirlpool, GOST-3411
• CMAC with AES and DESede
• CBC-MAC with AES, DESede, DES according to ISO/IEC 9797-1

• Key agreement with DH (Diffie Hellman) and ESDH (Ephemeral Static DH)

• Key-Pair generation for RSA (unlimited key-size), RSASSA-PSS, RSAES-OAEP, DSA, DSA-SHA2, DH, ESDH

• Random
  - NIST SP800-90 with SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, HMAC/SHA-1, HMAC/SHA-224, HMAC/SHA256, HMAC/SHA-384, HMAC/SHA-512, AES-128, AES-192 and AES-256
  - FIPS 186 with SHA 1, SHA 224, SHA 256, SHA 384, SHA 512, RIPEMD 160
  - ANSI X9.17
  - BSI AIS 20 (v2.0) E.5 with SHA 1, SHA-224, SHA 256, SHA 384, SHA 512, RIPEMD 128 /160, WHIRLPOOL

• Software key stores (PKCS#12, IAIKKeyStore)

• ASN.1 encoding and decoding (DER, BER, Base 64)

- X.509 Certificates
  - public key and attribute certificates
  - encoding and decoding
  - parsing and creation (issuing) of certificates
  - PKCS#7 and X.509 certificate lists
  - rich set of predefined extensions (PKIX, Netscape and more)
  - support for custom extensions
  - qualified certificates
  - all mandatory attributes of the PKIX AttributeCertificate profile and more
  - custom attributes

- X.509 CRLs with support for indirect and delta CRLs
  - support for efficient parsing of large CRLs

- OCSP (Online Certificate Status Protocol); client side and server side
  - direct support for transport via TCP and HTTP
  - OCSP response creation based on CRLs

- LdapURLConnection implementation for searching LDAP directories for certificates or attribute certificates and downloading crls from their distribution points

- Password Generator, Password Store and Password Strengh Checker utilities

- PKCS support
  - PKCS#1 version 1.5 signature and encryption
  - PKCS#1 version 2.1 PSS signatures and OAEP encryption
  - PKCS#5 password-based cryptography
  - PKCS#7 with support for single-pass stream processing
  - PKCS#8 private key information syntax
  - PKCS#9 selected object classes and attribute types
  - PKCS#10 certificate requests
  - PKCS#11 support via additional IAIK PKCS#11 Provider (separate product)
  - PKCS#12 personal information exchange syntax (private keys)

ECCelerate

ECCelerate is a toolkit which offers elliptic curve cryptography. It contains a CSP provider for the JCA/JCE framework of the Java™ platform. ECCelerate is based on Java 5/6 technology and offers easy to use elliptic curve cryptography, like ECDSA and ECDH, compliant with current standards. ECCelerate is an addon to the IAIK JCE provider (of version 4.0 or higher).

Feature List

- Compliant with ANSI X9.62-2005, ANSI X9.63, IEEE P1363a, FIPS 186-3, SEC1 v2.0, SEC2 v2.0 and RFC 5639
- ECDSA with SHA-1/SHA-2 support according to ANSI X9.62-2005 and BSI TR 03111 v1.11
- Fast finite field arithmetic in prime fields
- Fast finite field arithmetic in binary fields. In binary fields we only use polynomial base representation. This is mainly because of the patent situation, but there is no reason to use Gaussian normal bases.
- Support for elliptic curve arithmetic with affine and several types of projective coordinates
- Comprehensive domain parameter factory (ANSI X9.62, NIST, SEC2, Brainpool (RFC 5639) curves
- Support for elliptic curve arithmetic with affine and several types of projective coordinates
- JCE/JCA integration of ECDSA
- JCE/JCA integration of ECDH with and without cofactor multiplication
- ASN.1 encoding of signatures, public and private keys
- Speed-optimized
- Addon for supporting point compression and point decompression
- Support for Java Versions 5, 6, 7 and compatible (Please contact us, if you need support for older Java version, as, in such a case, we can offer you our old IAIK-ECC library.)

IAIK-PKI

PKI is one of the core activities at SIC/IAIK. Based on the knowledge we have earned by extensive research work in this area – for many years we have been partners of several European PKI-related projects – we have made PKI to one of the central parts of our Java Security software.
As a result of our research, we developed the IAIK-PKI module. This module represents a powerful und easy-to-use tool for certificate validation. Once configured, you need only to pass over the certificate you intend to validate. The PKI-module makes all necessary steps required for validation like retrieving CRLs, query OCSP servers and returns the result to you.The module can be used to enhance your applications about comprehensive, PKIX compliant path construction and certificate validation. Unique, compared to tools of its kind is the support for certificate and revocation information storage on remote databases. Designed to operate on thin clients as well as on heavy load servers, the IAIK-PKI module can store this necessary information on file systems and databases.

Feature List

- Certificate path construction using certificate repositories based on the file system, data bases, LDAP or a combination of these types.
- Certificate chain validation compliant to PKIX or ISIS- MTT .
- Handling of cross certification
- Automatic download of CA certificates according to the authority information access certificate extension.
- Handling of domain parameter inheritance of DSA and ECDSA keys
- Revocation checking based on CRLs and/or OCSP
- Archiving of CRLs
- High-performance caching mechanisms for CRLs and validation information
- Support for different kinds of certificate storage (Filesystem, Remote databases etc.)
- Customizable configuration mechanism. The module is delivered with an XML- and an property file base configuration mechanism. However, the customer can use it’s own configuration mechanism on demand.

IAIK PKCS#11 Wrapper

PKCS#11 specifies an abstract API for accessing cryptographic modules from the ANSI C programming language. This API contains functions for cryptographic operations like hashing, key generation, signature creation and verification, and encryption and decryption. As well, it provides some basic management functions for storage and retrieval of data objects like keys and certificates. Most vendors of smart cards and hardware security modules (HSMs) offer a PKCS#11 module for their hardware. There are even pure software PKCS#11 modules which aim at maximum performance through assembler optimized crypto routines.
Usually PKCS#11 modules are dynamic link libraries (DLLs on Windows or shared libraries on Unix) written in the C programming language. Unfortunately, it is impossible to access such modules directly from Java. This is due to the incompatibility of the Java Native Interface (JNI) with the PKCS#11 API.
The PKCS#11 Wrapper is a library to access PKCS#11 modules from the Java programming language. To fill the gap between JNI and PKCS#11, the wrapper uses a native adapter layer which maps the complete PKCS#11 API to a functionally equivalent Java API.

Feature List

- Supports PKCS#11 version 2.x
- Support for Java Versions 1.3, 1.4, 1.5, 1.6 and compatible
- Open-Source implementation under an Apache-Style license
- Well-designed and clean source code with complete JavaDoc
- Native part precompiled for Windows, Linux, Solaris and Windows CE 3.0, and simple to port to other platforms
- Developed based on PKCS#11 specification and not on specific products
- Vendor independent
- High Performance – the delay in the JNI adaptor is negligible
- Tested with many different smart cards and HSMs; e.g. Giesecke & Devrient, Utimaco, Oberthur, SeTec, Orga, IBM, Safenet, Schlumberger, Gemplus, Dallas, Rainbow, ActivCard, A-Trust, A-Sign, Eracom, Aladdin, Mozilla, Eutron, TeleSec, nCipher
- Tested and used in many other products (e.g. SUN JDK 1.5)
- Can be used in applets easily
- Comes with a rich set of samples

IAIK PKCS#11 Provider

The PKCS#11 Provider fills the gap between the PKCS#11 Wrapper and the JCA framework. This library contains a CSP for the JCA. Instead of Java implementations of the actual algorithms, this provider forwards all cryptographic operation requests to an underlying PKCS#11 module. For accessing the PKCS#11 module, this library employs the robust PKCS#11 Wrapper. To the application, the PKCS#11 provider behaves similar to a pure software JCA provider. Therefore, this product makes it easy to add support for smart cards and HSMs to existing Java applications, which currently only support pure software cryptography. Using the PKCS#11 provider requires little knowledge of PKCS#11. Typically, basic knowledge of Java and cryptography is sufficient.

Feature List

- Support for Java Versions 1.3, 1.4, 1.5, 1.6, 1.7 and compatible
- Supports PKCS#11 version 2.x compliant products
- Various predefined login dialogs – Swing Dialog, Swing Frame, Console
- Customizable to meet specific requirements–algorithms, login dialogs,...
- Hashes MD2, MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-128, RIPEMD-160, FASTHASH
- Signature schemes RSA PKCS#1 v 1.5 with MD2, MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-128, RIPEMD-160; RSA X9.31 with SHA-1; DSA; ECD SA; external hashing supported
- Key-Pair generation RSA, DSA, ECDSA, DH, DH/X9.42, KEA in hardware

- Symmetric ciphers AES, DES, Triple-DES (112 and 168 bit), RC2, RC4[TM], RC5, IDEA, CAST, CAST-3, CAST-128
  - encryption and decryption
  - key generation (directly in hardware)
  - mode of operation ECB and CBC
  - padding PKCS#5, no padding
- RSA encryption and decryption according to PKCS#1 version 1.5 and version 2 OAEP, ISO 9796, X.509 (raw)
- DH key agreement
- HMAC with MD2, MD5, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD-128, RIPEMD-160
- MAC with AES, DES, Triple-DES (112 and 168 bit), RC2, RC5, IDEA, CAST, CAST-3, CAST-128
- Hash-based key derivation in hardware
- Supports JCA KeyStore API, reading and writing
- True hardware random generation
- Based on robust PKCS#11 Wrapper
- Developed based on PKCS#11 specification and not on specific products
- Independent of PKCS#11 hardware vendor
- High Performance – suitable for server applications
- Tested with many different smart cards and HSMs
- Rich documentation with many code samples and FAQs

IAIK iSaSiLk

iSaSiLk is an implementation of the SSLv2 (client-side), SSLv3, TLS 1.0, TLS 1.1 protocols written in the Java™ language. It supports all standard cipher suites, including all AES and PSK cipher suites, and all standard TLS extensions. The initial version of iSaSiLk was one of the world-first Java™ SSL libraries. Today iSaSiLk is successfully used for securing numerous Web applications all around the world.

Feature List

- Implemented entirely in the Java™ programming language guaranteeing cross platform portability
- Supports JDK 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 and compatible
- Mature product with a proven nine year track record in real world applications
- Centralized security policy configuration
- Uses Socket API to allow easy upgrade of existing net work applications
- Support for the HTTPS protocol via the standard Java™ APIs
- Secures Java™ RMI
- Supports client side SOCKS and HTTPS proxies
- Multithreading safe
- Client and server implementation of SSL 3.0, TLS 1.0 and TLS 1.1; client implementation of SSL 2.0
- Ensures that the most secure protocol version and en cryption methods shared by client and server are used
- Supports all standard defined cryptographic algorithms including RSA, DSA, Diffie-Hellman, AES, Triple DES, DES, IDEA, RC2, RC4[TM], MD5, and SHA-1.
- Full strength, 40 bit export, and 56 bit export encryption.
- Supports all standard cipher suites, AES, Camellia, ECC and PSK cipher suites; easy plug-in of private cipher suites
- Supports all standard TLS extensions and Session Resumption without Server-Side state (SessionTicket extension)
- Public key server authentication, optional client authentication (RSA, DSA, and Diffie-Hellman) or fully anonymous connections
- Session caching for high performance connection establishment
- Security parameter renegotiation on demand
- Application Extensible Design
  - Can perform SSL over any pair of streams and over an existing socket
  - Pluggable custom certification path verification
  - Pluggable custom session management
  - Allows private application defined encryption methods
  - Allows private application defined compression functions

- Proven Interoperability
  - Interoperates with any SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1 implementation
  - Interoperability tested among others with clients Netscape, Mozilla, Firefox, Microsoft Internet Explorer, Opera.
  - Interoperability tested with servers from Netscape, Microsoft, Oracle, IBM, Apache (SSLeay, OpenSSL) and others.

- Cryptographic Provider Independence
  - Can be used with any JCA/JCE compliant cryptography provider
  - Can use several different cryptogrphy providers at the same time
  - Easy integration of Smartcards and other secure hardware devices
  - Comes with the IAIK JCE provider by default (included in the license)

IAIK-CMS with S/MIMEv3

Developers who want to write applications for signing or/and encrypting documents according to PKCS#7/CMS, or securing electronic mail according to S/MIME have to rely on program libraries offered by third parties. Neither PKCS#7, CMS nor S/MIME is supported by the JCA/JCE or by the MIME capable JavaMail^{TM} API. The JDK itself uses PKCS#7 utilities – for instance, when verifying a signed jar file. However, the corresponding libraries are only for private use and not publicly accessible.

The SIC crypto toolkits have supported PKCS#7 (included in IAIK-JCE) and S/MIME (version 2 implemented by IAIK-S/MIME) right from the beginning. IAIK-CMS succeeds the PKCS#7 and S/MIME libraries of IAIK-JCE and IAIK-S/MIME to implement the IETF standardized successor versions CMS and S/MIMEv3.

Feature List:

- Written entirely in the Java™ programming language guaeanteeing cross platform portability

- Works on JDK 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 and compatible
- Compatible with the javax.mail architecture from SUN

- Centralized security policy configuration

- Stream-based implementation for supporting one-pass processing

- Implements the IETF CMS, S/MIMEv3 and ESS specifications
- Implements all CMS content types Data, Signed-data, Enveloped-data, Digested-data, Encrypted-data, Authenticated-Data, Compressed-Data (RFC 3274) and Authenticated-Enveloped-Data (RFC 5083)

- Implements all CMS RecipientInfo types: KeyTrans-RecipientInfo, KeyAgreeRecipientInfo, KEKRecipientInfo, PasswordRecipientInfo (RFC 3211), OtherRecipientInfo (user pluggable)

- Supports all algorithms required and recommended for the implemented content types: SHA-1 (and also SHA- 224, SHA-256, SHA-384, SHA-512), MD5 (digest), RSA (signature, key transport), DSA, ECDSA (signature), X9.42 ESDH and SSDH, ECDH (key agreement), AES, Triple-DES and RC2 Key Wrap, HMACwith3DESwrap and HMACwithAESwrap (key encryption), AES, Triple-DES and RC2 (content encryption)
- Maybe used with any alternative algorithm implementa tion, fulfilling the requirements of the CMS / S/MIME protocols
- Supports Elliptic Curve Cryptography (ECDSA, ECDH)
- Supports DSA with SHA-2 according to FIPS 186-3
- Supports Camellia Encryption and Key Wrap (RFC 3657)
- Supports X.509 public key and attribute certificates
- Contains basic, extensible certificate verification and trust policy utility

- Supports all content types of S/MIMEv3: multipart/signed with application/pkcs7-signature, application/pkcs7-mime (signed-data, enveloped-data, certs-only, signed-receipt (ESS), compressed-data); + application/pkcs10 from S/MIMEv2

- Supports all Enhanced Security Services specified by ESS (RFC 2634, , 5035):
  - signing Certificates
  - security Labels
  - signed Receipts
  - secure Mailing List
- Supports ESS TripleWrapping and arbitrary nesting of S/MIME parts

- Application Extensible Design:
  - pluggable custom content-type implementations
  - pluggable custom certification path verification
  - pluggable custom cryptographic algorithm implementations
  - pluggable custom canonicalization (S/MIME) and security label policies (ESS)

- Proven Interoperability
  - interoperates with any CMS and S/MIMEv3 implementation
  - backwards compatible to PKCS#7v1.5 and S/MIMEv2
  - interoperability tested among others with clients Microsoft Outlook Express, Microsoft Outlook, Netscape and Mozilla Messenger
  - Listed in the IETF CMS Draft Standard Implementation Report

- Cryptographic Provider Independence
  - can be used with any JCA/JCE compliant cryptography provider
  - can use several different cryptography providers at the same time
  - easy integration of smart cards and other secure hardware devices
  - allows plug-in of user written JCA/JCE engines
  - allows plug-in of user written non JCA/JCE compliant crypto code
  - comes with the IAIK JCE provider by default (included in license)

IAIK-TSP

When verifying a signed document it is important to know when the signature actually has been created. Time values usually included in signed messages may only reflect the time the signer claims she/he has signed the document. For applying a reliable time stamp to the signature of a document, the PKIX working group at IETF has developed a special Time Stamp Protocol. The IAIK Time-Stamp Protocol API provides functionality and documentation for integrating a trusted time stamp authority or a time stamp client into an application properly.

Feature List:

- Written entirely in the Java™ programming language guaeanteeing cross platform portability

- Works on JDK 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7 and compatible

- Compliant with IETF TSP standard (RFC 3161)
- Implements all data types specified by IETF RFC-3161

- SigningCertificateV2 and ESSCertIDv2 support according to RFC 5816

- Encoding and decoding of time stamp requests

- Encoding and decoding of time stamp responses and accessing the contained information
- Implementation of TCP/IP, SSL and HTTP-based TSP clients
- Sample code of a TCP/IP, SSL and HTTP-based TSP server
- Cryptographic Provider Independence:
    - Can be used with any JCA/JCE compliant cryptography provider
    - Can use several different cryptography providers at the same time
    - Provisions for easy integration of smartcards and other secure hardware devices
    - Delivered with the IAIK-JCE provider by default

IAIK XML Security Toolkit (XSECT)

With the rapid propagation of XML in wide parts of Web technologies, the question of securing XML documents naturally arises. This development entailed the formation of a joint IETF/W3C standard for XML Signature Syntax and Processing as well as a W3C standard for XML Encryption Syntax and Processing.
We at SIC/IAIK have been - and still are - busy building our own XML crypto libraries. IAIK-IXSIL -- written by a member of the W3C/IETF XMLSignature working group himself -- has been one of the first XMLDSIG implementations around.
SUN also has initialized two Java Community Processes (JCPs) responsible for designing APIs for XML Digital Signature (JSR 105) and XML Encryption (JSR 106). SIC/IAIK has contributed to both efforts by participating in the respective expert groups.
The IAIK-XSECT library is a complete redesign of IXSIL incorporating both the XML DSIG and XML ENCRYPTION API.
As addon to IAIK-XSECT we procide a XAdES library supporting the ETSI XAdES standard for advanced XML signatures, which was also co-designed by SIC/IAIK.

Feature List

- Implemented entirely in the Java™ programming language guaranteeing cross platform portability

- Compliant to joint W3C/IETF standard „XML Signature Syntax and Processing“ (XMLDSIG) and W3C standard „XML Encryption Syntax and Processing“ (XMLENC)

- Creation/Verification of Detached, Enveloping and Enveloped Signatures
- Manifest and signature properties support both for signature creation and verification

- Support for all required and recommended XMLDSIG algorithms
    - RSA-SHA1 and DSA signature algorithms
    - HMAC with SHA-1 message authentication algorithm
    - SHA-1 digest algorithm
    - canonical XML canonicalization algorithm
    - XPath, EnvelopedSignature, Base64 and canonical XML transform algorithms

- Support for the additional algorithms
    - Base64, XPath, XPath Filter-2, Enveloped, XSLT Transform
    - inclusive and exclusive XML canonicalization (with and without comments)
    - Canonical XML 1.1
    - XPointers in the URI of an XML Signature reference element
    - RSA - SHA256, SHA384, SHA512, RipeMD160, MD5 signature algorithms
    - ECDSA - SHA256, SHA384, SHA512, RipeMD160 signature algorithms
    - HMAC with SHA-1, SHA256, SHA384, SHA512, RipeMD160, MD5 message authentication algorithm

- Support for all required and recommended XMLENC algorithms
    - AES128, AES192, AES256, TRIPLEDES block encryption
    - RSAv1.5, RSA - OAEP key transport
    - Symmetric key wraps: AES128_KW, AES192_KW, AES256_KW, TRIPLEDES_KW
    - Additional: ARCFOUR stream encryption

- XAdES addon: Support for advanced electronic signatures according to ETSI TS 101 903 V1.3.2, V1.4.1, V1.4.2 technical specification

The following chapter presents a selection of different fields of application for the IAIK Java Crypto toolkits:

Giesecke & Devrient and AET present the first Linux-based Smart Card Project

LVM (Landwirtschaftlicher Versicherungsverein Münster a.G.) is a nationwide all-round insurer for private customers as well as for small and medium-sized businesses, with a network of around 2,000 insurance agencies.
Certificates are used to protect data exchange between the stationary computers, notebooks and LVM's central server. This security system will allow the insurance company's employees to identify themselves quickly and reliably to the corporate network and to access uptodate information and program data at any time and anywhere. To gain access to the central server, employees use a smart card to authenticate themselves in both cases. Data exchange between mobile clients and central server is via GPRS (General Packet Radio Service) or, in the case of stationary clients, LAN.

The PCs in the agencies and the notebooks of the field staff run with the Linux operating system and the LVM-specific application LASII.

The main components of this Public Key Infrastructure are the StarSign ® Token-solution of Giesecke & Devrient which is based on the chip card operating system STARCOS SPK 2.3.
Another essential component is the IAIK Java wrapper which functions as the interface between the LASII Java application (which does not support PKCS #11 for token integration) and the Safe Sign PKCS# 11 Library of AET Europe.
Through this solution, users can authenticate to the application by means of certificates on a smart card.

AGFA

AGFA uses IAIK products to cover the secure communication aspects and to be compliant with HIPAA in the following products: Drystar Medical Printer, PAXPORT (a DICOM gateway).

Certipost

Certipost enables organisations to communicate electronically with any customer, citizen, supplier and public institution, by automating inbound and outbound information flows, streamlining document exchange and by securing and certifying electronic communications. Furthermore, Certipost is supplier of the digital certificates within the Belgian electronic identity card (eID). Certipost offers solutions enabling electronic invoicing, document exchange within the supply chain, e-government with social security and customs, electronic registered mail, electronic security, electronic counters and the use of the electronic identity card within organisations. In order to secure all these solutions, Certipost has selected IAIK Java Toolkit.

CURIAVANT

Curiavant Internet GmbH was founded in 1999 by the cities of Nürnberg, Fürth, Erlangen, Schwabach and Bayreuth. Prior those cities won the nationwide MEDIA@Komm contest with their concept of a digital town hall. During the last years Curiavant has put this concept into practice. The heart of the company services is the development of the secure and legally binding e-government infrastructure CuriaWORLD as well as of municipal webapplications using the technology of electronic signatures. For the first time Curiavant implemented web-applications that allow official online communication without discontinuity of media, such as printouts for signing personally.
CuriaWORLD acts as interface between the user front end and the public administrations back end systems. In our products CuriaWORLD and CuriaSIGN, we use the IAIK tool kit as cryptographic provider, especially for the use of certain encryption mechanisms within the electronic signature. With the IAIK-software we realize interoperable und secure infrastructure applications for the users of electronic signatures in business and public administration.

Entegrity`s ASSURE ACCESS

AssureAccess uses an embedded PKI structure to provide a trust infrastructure allowing AssureAccess components to securely exchange information. The same technology enables AssureAccess to support certificate revocation mechanism and certification discovery. In addition AssureAccess builds upon Stiftung SIC software to provide cryptographic support for Security Assertion Mark-up Language (= SAML - a technology enabling cross domain single sign-on).

Ergon

Ergon uses iaik in its internet banking projects where it allows the Java client application to communicate with the server in a secure way. These projects are

- DirectNet (internet banking solution of Credit Suisse)
- the internet banking solution of Liechtensteinische Landesbank
- the Java application for the external asset managers of Credit Suisse
- the Java Client in the order routing monitor of Credit Suisse

Ergon Informatik AG (www.ergon.ch), founded in 1984 and employing 65 people, is a leading, independent Swiss software engineering company specialized in design, development and implementation of high sophisticated software solutions based on open systems. As a center of Java expertise and as the Swiss pioneer of mobile application development, Ergon is a true authority. Major customers of Ergon are Credit Suisse, UBS, Rentenanstalt/SwissLife, Swisscom, Sunrise, Coop, Swiss Federal Railways (SBB), Roche and Novartis.

Esmertec

Esmertec is a leading provider of Java™ solutions for mass market devices such as mobile multimedia phones, personal digital assistants (PDA) and home multimedia systems. Our high performance and quality J2ME solution includes Jbed™ CLDC, Jbed CDC and Jcap™. Esmertec is using IAIK iSaSiLk ME / JCE ME for MIDP2.0 on mobile phones.

Guardeonic`s TrustedSigner

Secure Document Related Technologies (SDRT) as provided by Guardeonic Solutions AG is a major enhancement to existing Document Related Technologies solutions, since the „Secure“ in the DRT provide a legally liable solution for electronic documents. Thus old paper-based processes can be replaced completely by electronic processes. This leads to a considerable process and cost optimization. DRT involve a wide range of IT security aspects, e.g. encryption, digital signatures, ecurity policy, Public Key Infrastructure (PKI) and smart cards. Guardeonic Solutions AG provides the long-year security experience to meet the technical and legal requirements to SDRT. Guardeonic Trusted Signer is a SDRT enhancement for existing DRT solutions. TrustedSigner meets legal regulations to qualified digital signatures and is going to be certified according to Common Criteria EAL3+.
TrustedSigner will also be integrated into Livelink® by Open TextTM Corporation - the market leader in Document Management Solutions.
Major Guardeonic SDRT projects include the German electronic income tax return solution „ELSTER“ and a document management, digital archiving and records management solution for the University Hospital of Tübingen.

Lucent Technologies NavisRadius Server

The Lucent Technologies NavisRadius Server provides Authentication, Authorization, and Accounting (AAA) services to network access devices such as remote access dial equipment, Wi-Fi Access Points, and 3G mobile network data systems.

NavisRadius is a high performance, scalable server based on a flexible and extensible Java based Policy-Flow plug-in architecture.
NavisRadius includes cryptographic software from Stiftung SIC to support its Extensible Authentication Protocol (EAP) services.

MOA: Modules for Online-Applikations

In Austria MOA-modules are available for free for the use within public administrations, but they can also be purchased for commercial use. The MOA-modules are based on the IAIK-toolkit suite, and they can be used to create secure online procedures emulating conventional administrative procedures electronically; several basic services related to electronic signatures are required.

Electronic Signature of citizens:
Verifyabilty of requests regarding content and origin by the authority (who filed the application? Has the request not been tampered with?)

Electronic Signatures of the authority:
Verifiability of the authenticity of documents created by an authority regarding contents and origin by a citizen or another authority (Who created the document? Has the document not been tampered with?)

Authentication of citizens:
Verificaton of the identity of a citizen, contacting an authority using an onlince-process (who is communicating with you).

Swiss Post

The Swiss Post – Information Technology Services Tessin - has introduced the virtual postal gate on the market actively using J2EE‘s technology and the IAIK Javabased crypto products of the Stiftung SIC in Graz/Austria.

The secure virtual postal gate introduces the integration of mail and other services, like E-Mail Letter and Address Change Online while using a mouse only (https://www.postmail.ch).

SIC Crypto Libraries im TrustSuite TCM des TÜV Süddeutschland

TÜV Süddeutschland has set new benchmarks in the fields of high-quality securing of critical data processing processes - the development of the TrustSuite Trust Center Modules (TCM), in particular for key management and the personalization of smart cards. The innovative system design and the new ways in the smart card mechanics of transport form a universal platform, on which highly flexible, compact and inexpensive safety solutions can be presented by means of appropriate Crypto software.
The necessary high quality of the cryptographic operations is made available by the Java Cryptography Extension set of Stiftung SIC (IAIK JCE). Almost all functions installed in the TCM which are responsible for the production of the cryptographic key material and the corresponding certificates, profit from the flexibility and the high functionality of this toolkit, which also co-operates without problems with the assigned hardware crypto modules.
Abstracting the access to the smart card by means of the Java PKCS#11 Wrapper provided by IAIK JCE, a very flexible interface of the smart card could be realized.
On contrary to client applications, this interface also relieves the developer from time-consuming routine tasks during write accesses within the process of personalization.

www.trustsuite.de

Tumbleweed's SecureTransport

Tumbleweed Communications is a leading provider of secure Internet messaging solutions used by many of the world's leading banks, insurers, healthcare organizations, government agencies and Global 500 companies to securely and reliably link their back-end applications, remote offices, customers, partners and suppliers.
By making Internet communications secure, reliable and automated, Tumbleweed's email firewall, secure file transfer, secure email, and identity validation solutions help customers significantly reduce the cost of doing business. Tumbleweed's SecureTransport sets the standard for multi-protocol and multi-client secure file transfer over the Internet and private IP networks.
SecureTransport's enterprise-class features, such as strong authentication, automated batch transfers, reliable delivery of very large files, guaranteed delivery and data integrity, secure DMZ streaming, extensive logging and legal-grade auditing, and event-driven agents for rules-based application integration make it a critical component in many business integration processes,
financial and healthcare networks, and trading communities. Tumbleweed products are used by millions of end-users and tens of thousands of corporations.

XPS Software - Trex and JProtector

TRex
TRex is a combination of a secure bi-directional tunnel via TCP/IP and a Java application programming interface.
TRex allows the execution of remote transactions on the IBM operating systems VSE/ESA, OS/390, zSeries and iSeries (AS/400), controlling it from a java platform.
All necessary cryptographic functions are executed by the IAIK Crypto Toolkit. Symmetrical and asymmetrical coding are used (Blowfish, RSA), as well as service methods for digital X.509 certificates, and methods for the use of PKCS#12 objects.

JProtector
JProtector is a Java-based 3270/5250 terminal and printer emulation. Apart from the use as telnet emulation, JProtector can also be used as emulation for the protected online access to IBM host computer. For the realization of safe connections JProtector makes use of the TRex technology described above.

XiCrypt - Technologies

XiCrypt's mission is to develop, implement and validate solutions for a global networked economy in which workers, organisations and governments interact easily and dynamically through a secure, trustworthy and reliable ubiquitous information and communications technologies infrastructure.
S/Mime Gateways
S/Mime Mapper by XiCrypt Technologies provides an easy, centralized, secure and const-efficient way to secure your email transactions. No user training or changes in the email infrastructure are needed! Central policies allow for their easy enforcement. Encryption guarantees the confidentiality of your data. Define your own policies: You can sign or encrypt any email to whatever domain or recipient. Users do not have to care about encryption and digital signatures. S/Mime Mapper decides in the background which emails have to be signed and encrypted.
Compatible to all important Crypto-Hardware-Products, Email-Systems (Exchange, Lotus Notes, Groupwise, sendmail, postfix, Outlook) and standards (S/Mime Mapper is conformant with standards like SMTP, POP, IMAP, PKCS11, PKIX, X.509, TSP, XAdES, S/Mime, OCSP, PGP, LDAP, ...)

eBilling Solutions - Legally binding electronic invoicing eBilling by XiCrypt is based on S/Mime Mapper and provides both transmission and long term secure storage of electronically signed invoices by using time stamping and XAdES technologies. The solution of XiCrypt supports all kinds of signatures in both sending and storing documents. As it is based on the S/Mime Standard, it can be used with any software in charge for bookkeeping. The recipient of an invoice has only to have access to a standard email client. eBilling by XiCrypt integrates easily with used systems like SAP (including iDocs), Navision or BAAN.

worldwide
business

Please contact us as following for:

sales-related information	.... jce-sales@iaik.tugraz.at
general technical information and support (evaluation versions and betas)	.... jce-support@iaik.tugraz.at

SIC

Stiftung Secure Information
and Communication Technologies
Inffeldgasse 16a
A-8010 Graz
Austria
Phone: +43 316 873 5582
Fax: +43 316 873 5593
https://jce.iaik.tugraz.at