goldbug-manual

Manual of the GoldBug Crypto Messenger

View on GitHub

GoldBug-manual

Manual of the GoldBug Crypto Messenger

https://compendio.github.io/goldbug-manual/

& https://compendio.github.io/goldbug-manual-de/

Edited by Scott Edwards (2018, Review at Github).

(A first drafted translation into English by a translation machine).

Content

Manual-Files: https://github.com/compendio/goldbug-manual

Manual Deutsch/German: https://github.com/compendio/goldbug-manual-de

PDF: https://sf.net/projects/GoldBug/files/goldbug-manual-de.pdf

Website: http://goldbug.sourceforge.net/

Download: https://sourceforge.net/projects/goldbug/files/

Source: https://github.com/textbrowser/spot-on

1 What is GoldBug?

GoldBug is a secure instant chat messenger and encrypting email client that also include additional features such as group chat, file transfer, and a URL search based on an implemented URL database.

Thus, the three basic functions frequently used by a regular Internet user in the Internet - communication (chat / e-mail), web search and file transfer - are represented in an encrypted environment safely and comprehensively.

In addition, GoldBug has also implemented a number of useful tools, such as encrypted chat server functionality, proxy-enabled passthroughs, text / ciphertext conversion pads and vice versa, a feedreader and web crawler, or dashboards for the friends of statistics and analysis, and much more.

With the use of GoldBug - GB for short - the user can therefore be relatively sure because of the encryption that no unwanted third party can eavesdrop on the conversations or open emails or file transfers. The URL search also happens on the local machine, so that search queries are protected and secured.

The user-to-user communication should remain with this application via the Internet in private, protected space.

GoldBug uses for this strong and multiple encryption, also called hybrid encryption, with different levels of modern encryption technology based on established encryption libraries -

such as libgcrypt (known from OpenPGP or GnuPG and OpenSSL).

For example, it creates separate and different public / private keys for encryption and signatures for each function - based on the encryption algorithms RSA, or alternatively Elgamal and NTRU. In addition to NTRU, the encryption algorithm McEliece is open source implemented.

These latter two algorithms are considered to be particularly secure against attacks that are known from quantum computing and are becoming increasingly relevant in the future due to fast quantum computers. GoldBug is thus one of the first messengers worldwide to implement these two algorithms, thus initiating the renunciation of - or alternatives to - the RSA algorithm - that has been officially considered as broken since 2016 (see NIST cited in Adams / Maier 2016).

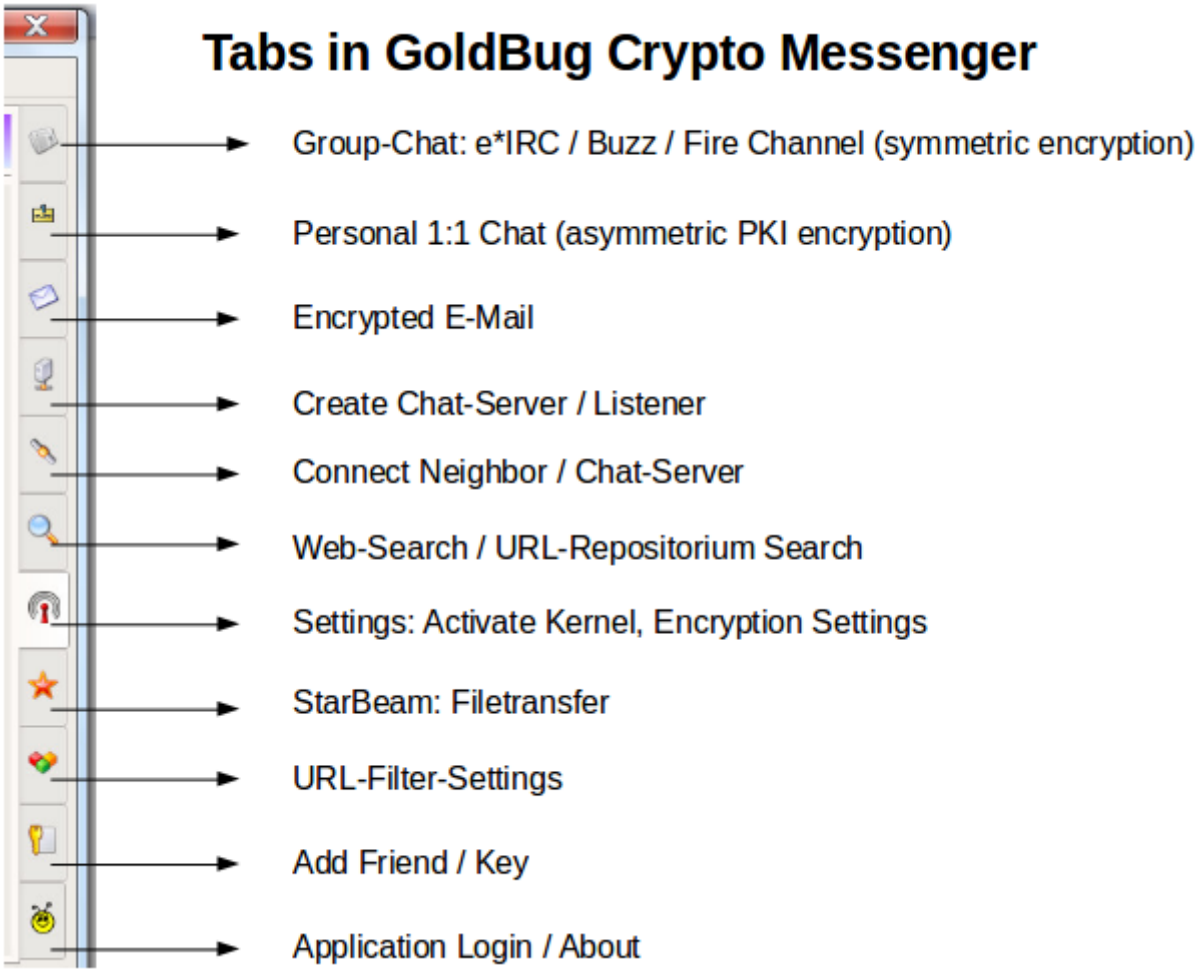
Furthermore, the application also offers decentralized and encrypted e-mail as well as decentralized public group chat in IRC style. Finally, there is also the function to implement a URL web search in a decentralized database repository: Users can store URLS and content of the website - as so far Bookmarks in the browser and its cache - in a comfortable searchable database to their thematic RSS feeds or import them, which is based either on SQL or PostGres and is also p2p networkable.

The email can be IMAP, POP3 and thirdly, P2P e-mail - GoldBug is thus a fully functional e-mail client. As soon as encrypted emails are sent, it is necessary that the friend also uses this (or any other echo) client. This has the advantage that the encryption key is only to be exchanged once, but then no longer has to be applied to every single e-mail. This function of transferring the key encrypted back in a direct way is called “Repleo” in GoldBug and later on this was also taken over in other projects under the name Autocrypt or KeySync (within an automatic process).

As in any messaging program, files can be shared and sent. The transfer is always encrypted per se. With the tools “Rosetta CryptoPad” and the “File-Encryptor” the user can additionally encrypt text and/or files additionally safely or convert them back. These encryption tools can therefore also be used for other transmission paths (such as an unencrypted path outside of GoldBug).

With all its equipment, GoldBug is therefore a so-called “Communication Suite” - a program with numerous functions for secure communication, which realizes the transmission of the encrypted packets with the so-called echo protocol. This is particularly secure, as it will be explained below.

Figure: Explanation of the tabs in GoldBug Crypto Messenger



1.1 Why is it important for Internet users to encrypt communications

Today almost all wireless Wifi Internet accesses are password protected (So called “Freifunk”-activities are currently trying to reverse this over-regulation through password-free and account-free wireless Internet access). In a few years plain text messages or e-mails to friends (in the following all terms always apply to all genders) via the internet should be encrypted as well. In order to consolidate this change, sometimes C-Mail (for crypto-mail) rather than e-mail is used as a new term.

Encryption is not a question of having something to hide or not, it is the question of whether we ourselves control our communication - or whether it is controlled by others, third parties.

It is ultimately also a question of attacking free thinking and a question of deleting the presumption of innocence (“In doubt for the accused” - if every citizen ever belongs to a dock!).

Democracy requires thinking and discussing alternatives in private as well as in public.

The communication and data transmission over the Internet should be protected as parents would also protect their loved ones or a mother bird their young against the unknown: Everyone should protect his privacy and human rights with modern encryption functions.

Strong multi-encryption (so-called “hybrid encryption”) thus ultimately secures the declarations of human rights in their broadly constituted consensus and is a digital self-defense that

everyone should learn and use - to ultimately contribute to democracy and support them.

The GoldBug Messenger tries to be an easy-to-use tool for this claim. Similar to the development of safety in automobiles, the e-mail & chat encryption will also develop: if you initially drove without a seatbelt in the car, today we drive with obligatory safety belts (e.g. since 1968 in the U.S.) and additional airbags or thirdly additional electronic security information systems.

GoldBug is an easy-to-use, but to some extent also learnable program; it requires - as with the car driver's license - the knowledge of the various controls and options. GoldBug is thus an already simplified user interface compared to the original user interface, which is also called Spot-on coming from the "Spot-on" project. Similar to a cockpit of an aircraft, there are even more control buttons available in this original user interface. In GoldBug these are already reduced and also another minimal view is offered for beginners in software for cryptographic processes. In this respect: Learn what is still unknown and note that it is already a reduced scope. This manual can help you to understand the individual functions. And users who first read and then try out are, as always, clearly in the advantage. :-)

The unencrypted Plain Text email or chat message should therefore have actually become obsolete after the 2013 Snowden Papers found that private emails are widely intercepted and systematically collected and evaluated by many interested parties worldwide.

Incidentally: The logo of the GoldBug logo is written in the font "Neuland" (which means translated: new territory) - a font that was developed in 1923 by the typographer Rudolf Koch. Interestingly enough, the logo has been an allusion to the German "sentence of the year" 2013, when German Chancellor Angela Merkel - in connection with the surveillance and espionage affair in 2013 and the Listening to her personal mobile phone - in a conversation with American President Barack Obama coined the phrase: "The Internet is a new territory for us all." ..

.. - How long encryption for the subsequent student generations will remain a new territory (or " Neuland ") or literally 'secret science' - or a kind of "seat belt", which will also convert e-mails to c-mail, decide the learners, teachers and the media and technicians - but in any case everyone (e.g. the reader of this manual) with the own friends with whom this or other encryption software is used.

Figure: GoldBug logo (smiley face)

1.2 Where does the name “GoldBug” come from?

The Gold Bug is a short story by Edgar Allan Poe: The plot is about William LeGrand, who recently encountered a gold-colored ladybug.

His buddy, Jupiter, now expects LeGrand to evolve in his quest for insight, wealth, and wisdom after being in contact with the Golden Bug - and thus goes on to another friend of LeGrand, a narrator not further mentioned by name, who thinks it would be a good idea to visit his old friend again. After LeGrand then encountered a secret message and was able to decrypt it successfully, the three start an adventure as a team.

The Gold Bug - as one of the few pieces in the literature - integrates cipher text as an element of the short story. Poe was thus far ahead of the popularity of cipher texts of his time when he wrote “The Gold Bug” in 1843, in which the success of history turned to such a cryptogram and metaphorically to the search for the knowledge of the philosopher’s stone.

The Gold Bug was a much-read story, extremely popular and by the literati the most studied work by Poe during his lifetime. His ideas also helped to promote the writing of encrypted texts and so-called cryptograms (see also Wikipedia).

Over 170 years later, encryption in the Internet age has more weight than ever. Encryption should be a standard when we send communications over the insecure internet - reason enough to use this name for the application to remember the origins of the encrypted writing.

GoldBug is thus a historical tribute, which possibly requires an adaptation to the term, because “bug” is often understood in the IT language as an error correction. Depending on the person, the idea of valuing a golden ladybug as much as another cuddly toy may require a strong cognitive reorganization of a so far dominated worldview or the routinized expansion of the appreciation of bug-finds as interesting research finds.

Those who like exploring new things, openly approaching what is found, will be able to learn and deepen many things in cryptographic processes with the GoldBug application, if so far no access to this “new territory” has been made possible. For teachers, the software is therefore an interesting teaching tool that can introduce and test cryptography in practical implementation and exercises with playful testing, reminiscent of the beginnings of popular cryptography at the time.

2 Encryption: GoldBug has alternatives to RSA - The first NTRU & McEliece Messenger

Encryption is only as good as the mathematical calculations cannot be calculated by the automation of computers at lightning speed. Therefore mathematically speaking, the prime factorization is used because it requires years of computational effort.

There are basically two methods of encryption. First, the symmetric encryption: Both users use the same password, e.g. a so-called AES with 32 characters, which will be explained in more detail below, or on the other hand, there is the a-symmetric encryption. In a-symmetric encryption, each user has two keys, a private and a public key. The users each exchange the public key and can then encrypt data using the private key in combination with the public key. After transmission, the other party is also able to decipher the message with their own private key. The asymmetric method is also called PKI: Pubic Key Infrastructure called, which can build on different algorithms for key generation.

However, encryption - be it via AES or PKI - is not unbreakable, and the procedures and libraries must also be well-used to be secure. RSA is considered “today as an essential, widely studied

and not yet breakable encryption standard - although the further development of fast computers might bring a different future", - it was still 2014 in this manual noted. In 2016, the official NIST Institute announced that RSA is considered broken in the age of quantum computing (see NIST cited in Adams / Maier 2016).

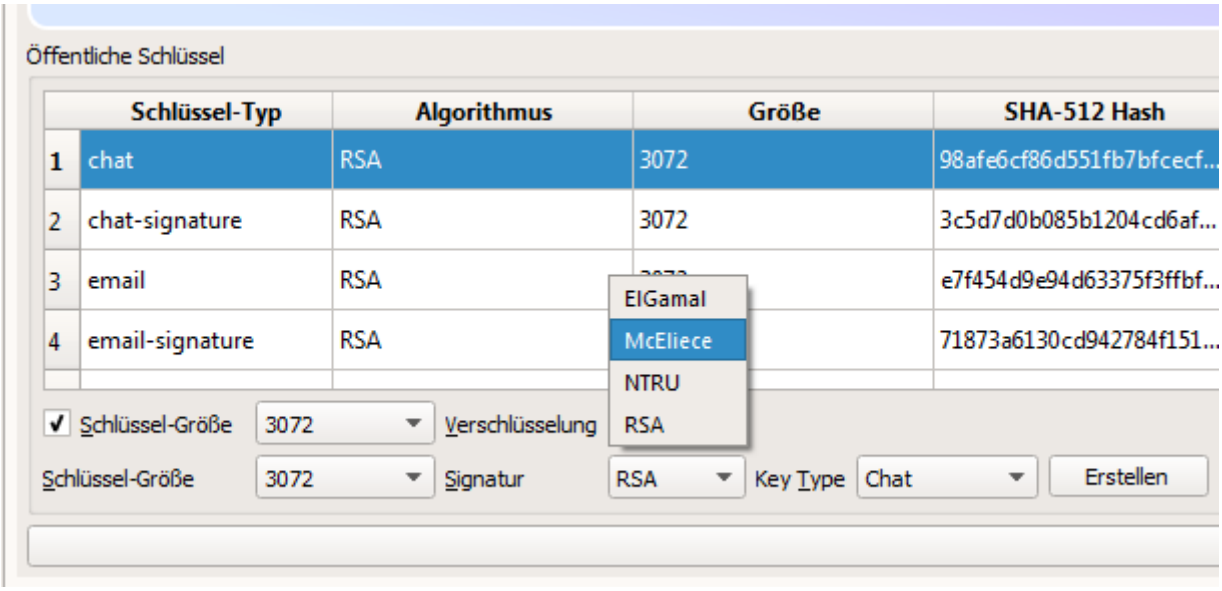
The media has barely picked it up, as everyone will probably agree that you cannot buy a quantum computer in the nearest supermarket, so the problem might be not relevant. It has the charm of children who hold their hand in front of their eyes and thus do not let the problem or risk endanger their perception of reality. Nevertheless, it is officially confirmed that RSA can be broken - with special means. The security is gone. This also has an impact on our Internet economy and online banking, because so far, the secure connections are relying on RSA and a TLS to secure the connection to online banking or shopping based on McEliece or NTRU is not yet developed.

2.1 Asymmetric encryption with PKI: RSA, Elgamal and especially NTRU and McEliece in comparison

Therefore, GoldBug Messenger has already introduced additional alternatives to RSA at an early stage - if this encryption algorithm standard would ever be insecure: RSA with a correspondingly large size of the key (at least 3072 bytes) can still be regarded as a time hurdle by non-specialized technical administrative staff, especially as GoldBug holds more extensive security even with multiple encryptions.

In addition to RSA GoldBug has yet implemented the encryption algorithms Elgamal and also NTRU and McEliece. The latter two are also considered to be particularly resistant to the attacks known from quantum computing.

Figure: McEliece's algorithm for advanced protection against attacks from quantum computing



GoldBug uses the libgcrypt, libntru, and McEliece libraries to create persistent private and public key pairs. Currently, the application generates key pairs for each of the ten functions during

initialization. Key generation is optional. As a result, GoldBug does not require public key infrastructure. Of course, the desired algorithms can be selected and keys can be generated.

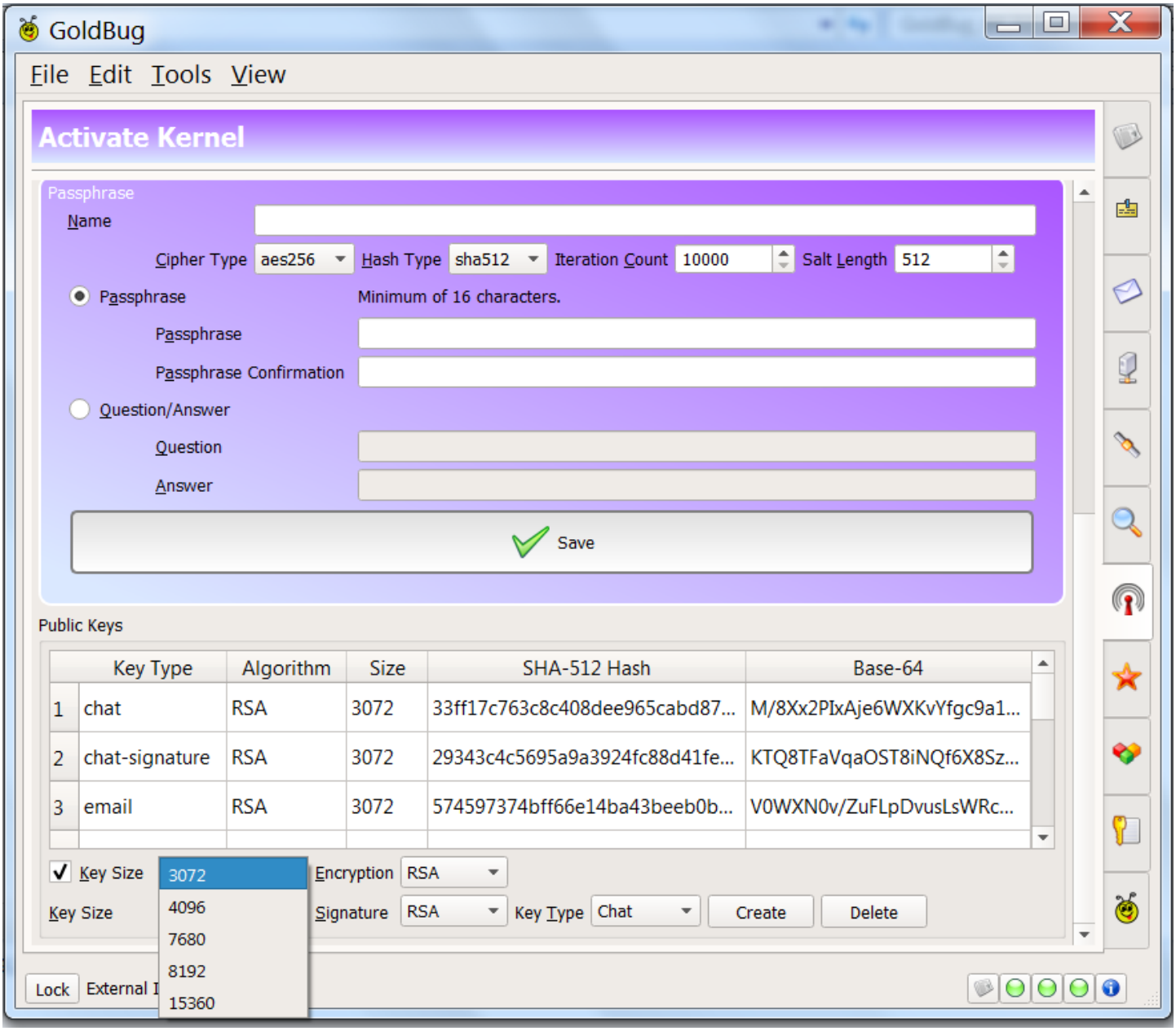
There is also a comprehensive choice of encryption methods available with encryption: DSA, ECDSA, EdDSA, Elgamal, and RSA. Signature means that the generated encryption key is re-signed with a key to prove that a message is coming from a particular subscriber and nobody else. The OAEP and PSS schemes are used with the RSA encryption and RSA signature respectively.

Figure: RSA and its alternatives in GoldBug

McEliece-Kryptosystem
The McEliece cryptosystem is an asymmetric encryption algorithm. It was presented in 1978 by cryptographer and founder Robert J. McEliece. Even with the use of quantum computers, there is no known efficient method by which the McEliece cryptosystem can be broken. This makes it a promising algorithm for post-quantum cryptography.
NTRU
NTRU is an asymmetric encryption technique developed in 1996 by mathematicians Jeffrey Hoffstein, Jill Pipher and Joseph Silverman. It is loosely based on lattice problems that are considered unbreakable even with quantum computers. However, NTRUEncrypt has not been extensively studied so far as more common methods (e.g. RSA). Ntruencrypt is by IEEE P1363.1 standardized (see Ntruencrypt)
Elgamal
The Elgamal encryption method or Elgamal cryptosystem is a public-key encryption method developed in 1985 by the cryptographer Taher Elgamal, based on the idea of Diffie-Hellman key exchange. The Elgamal encryption method, like the Diffie-Hellman protocol, relies on operations in a finite-order cyclic group. The Elgamal encryption method is provably IND-CPA-safe, assuming that the Decisional-Diffie-Hellman problem is not trivial in the underlying group. Related to the encryption method described here (but not identical with it) is the Elgamal signature method (the Elgamal signature method is not yet implemented in GoldBug). Elgamal is not subject to a patent (cf. Elgamal encryption method).
RSA
RSA (after the people Rivest, Shamir and Adleman) is an asymmetric cryptographic procedure that can be used for both encryption and digital signature. It uses a key pair consisting of a private key used to decrypt or sign data and a public key to encrypt or verify signatures. The private key is kept secret and can only be calculated from the public key with extremely high expenditure (see RSA Cryptosystem).

GoldBug encryption is designed so that any user can communicate with any user, no matter what encryption algorithm a user has chosen. Communication between users with different key types is thus well defined when the nodes share common versions of the libgcrypt and libntru libraries: anyone who has chosen an RSA key can also chat and email encrypted with an user who has chosen an Elgamal key. This is because everyone supports each algorithm and the library supports it. If you want to test the program with a friend, it is best to use the latest version of GoldBug.

Figure: Customizable crypto



Of course every user in GoldBug can set his

- individual key size,
- the “cipher”,
- the “hashtype”,
- furthermore “iteration count”,

- and the cryptographic salt length .. which are often typical parameters for key creation and encryption.

The advantage is that every user can define this individually for himself. Other applications - even open-source applications - hardly create for the user this choice, to determine these key values for the encryption process itself.

2.2 Application of Block Cipher Modes & Encrypt-then-MAC

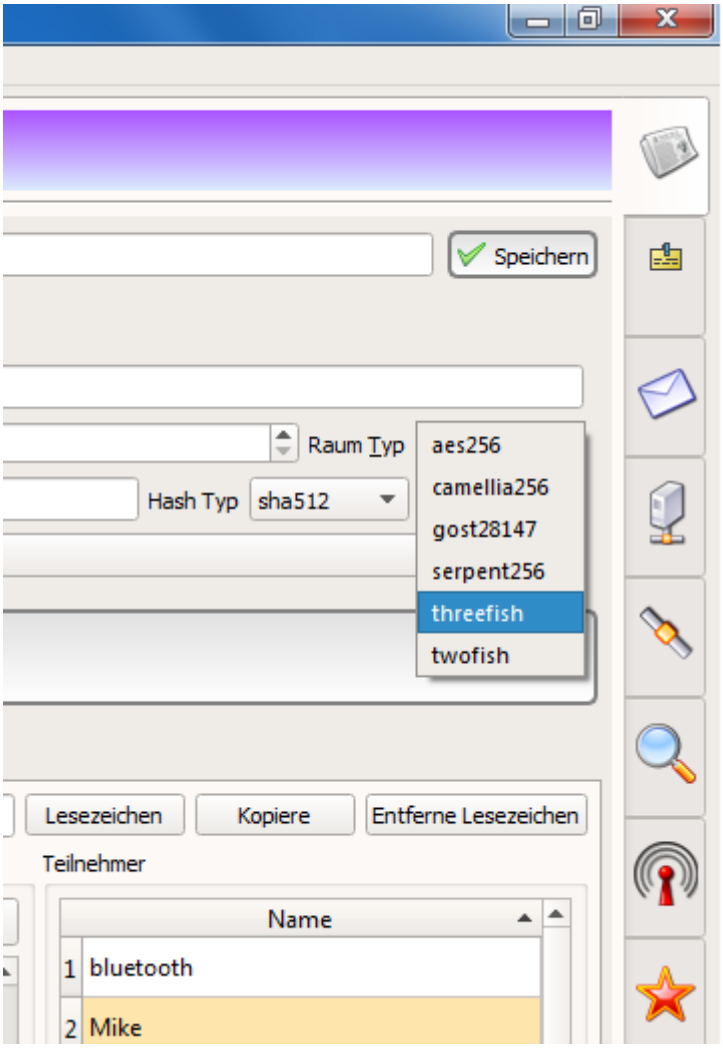
GoldBug has not only standardized forward-looking algorithms or numerous details such as the switch from AES-128 to AES-256 or the use of very high, because necessary key sizes, but also implemented the professional integration of established and new encryption processes.

GoldBug uses CBC with CTS to provide confidentiality. The file encryption mechanism supports the Galois / Counter Mode (GCM) algorithm without the authenticity property provided by the algorithm. To provide authenticity, the application uses the methodical approach of “Encrypt-then-MAC” (ETM). MAC stands for Message Authentication Code - and means that the order is determined: first encrypt, then authenticate the message with a code. The documentation for the source code for the section of encrypted and authenticated containers contains further technical details.

Another example of this innovation is the implementation of the ThreeFish hash, which was available as an alternative to SHA-3 when it was realized that SHA-1 was no longer able to cope with future requirements.

Threefish is a block encryption developed as part of the design of the cryptographic hash function Skein, which participated in the NIST selection process for SHA-3. Threefish does not use S-boxes or other lookup tables to complicate time-side attacks (computing time attacks).

Figure: Theefish implementation



Many more examples - especially compared to other messengers - can be found, which prove that the encryption processes in GoldBug are very state-of-the-art.

2.3 Hybrid encryption system

GoldBug implements a hybrid encryption system, including authenticity and confidentiality. Hybrid means first of all: “both variants are available” and can be combined with each other. Thus, a message can first be a-symmetrically encrypted with PKI shown above and then symmetrically with an AES again. Or the other way round, there is also another variant conceivable. The PKI transmission path transmits with permanent keys again only temporarily used keys, with which then the further communication takes place over this temporary channel. The temporary channel can then again transmit a symmetric encryption with an AES.

Thus, not only in the method change from PKI to AES respective from asymmetric encryption to symmetric encryption exists one option to build a hybrid system, but also in the switch from permanent PKI keys to temporarily keys.

Encrypting often and switching between these methods or time-limited keys is a strong competence of GoldBug in this multiple and hybrid encryption. With these possibilities you can now play and apply them in various ways. Is the permanent or the temporary key applied first,

or once again the symmetric and then the asymmetrical as the second level of encryption? or vice versa?

Part of the system in GoldBug generates the key for authentication and encryption per message. These two keys are used to authenticate and encapsulate data (that is, the message). The two keys (for authentication and encryption) are then encapsulated across the public-key part of the system. The application also provides a mechanism for distributing session keys for this data encapsulation (or encryption of the message) as described above, the temporary key. Again, the keys are encapsulated and transmitted via the public key system: an additional mechanism allows the distribution of the session keys over the predetermined keys.

As an example, this format may serve the following message encryption:

```
EPUBLIK Key
(Encryption Key || Hash Key)
|| EEncryption Key (Data)
|| HHash Key (EEncryption Key (Data)).
```

For those who are dealing with encryption for the first time, the above example of encapsulation is a first example to further study and understand the methods; - In any case, you can see how the encryption key is supplemented by the hash key (see MAC) and also the data is embedded in different encryption levels.

Non-NTRU private keys are evaluated for correctness by the gcry_pk_testkey () function. The public key must also meet some basic criteria, such as the inclusion of the public key identifier.

The authentication of the private key and the encryption mechanism is identical to the method as further discussed in the documentation of the source code in the section on the encrypted and authenticated container.

However, let’s take a simpler case and go into more detail about symmetric encryption with an AES password, which can complement PKI encryption as follows:

2.4 Symmetric encryption with AES

Symmetric encryption uses AES - a 32-character password generated by random processes. Since all characters and special characters are used in the generation, the set of options is also sufficiently large that even fast machines can not try out all variants within a short time. While asymmetric encryption uses a public and private key pair, in symmetric encryption it is a secret passphrase that both subscribers need to know (hence called symmetric) (- or for GoldBug: in the later discussed Gemini function it is called “Gemini” (from the Greek term for “twin” derived): Both sides have to exchange and know the secret passphrase).

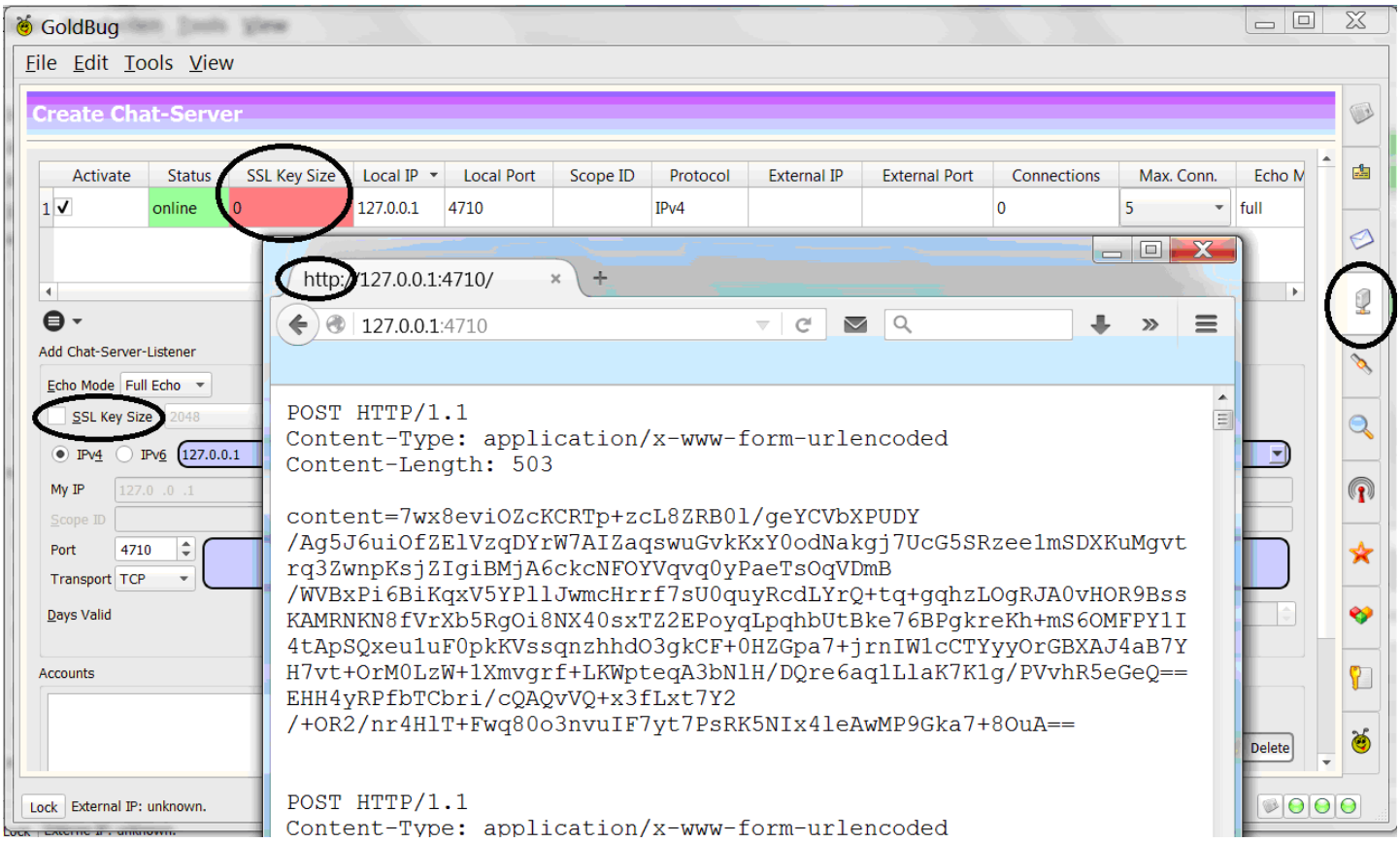
GoldBug thus uses both standards as described above: asymmetric keys and/or symmetrically encrypted messages are sent by SSL/TLS (i.e. asymmetric) encrypted connections, and the asymmetrically encrypted message can possibly also be secured with symmetric encryption (AES), Then GoldBug even uses three levels of encryption like this example of encapsulation (simplified, as shown without HASH / MAC or signature):

```
RSA-SSL/TLS (AES (Elgamal (Message)))
```

Translation of this formula: First, the text message is encrypted with the public (asymmetric) key of the friend via the Elgamal algorithm, then the encrypted text is encrypted again with an AES algorithm (symmetric password) (a second time) (and secured) and this capsule is then sent to the friend through the existing SSL/TLS (using RSA) encrypted (asymmetric) connection.

If an HTTP listener is set up and the encrypted message capsule is not sent via HTTPS - over the third encryption layer, via an SSL/TLS connection - the cipher text of the message capsule can also be viewed in the browser. It turns out that even with two encryption layers only cipher test is sent (see illustration from the practice demo of Adams / Maier 2016).

Figure: ciphertext



It is also possible to exchange the symmetric passphrase (the AES) with the remote station using established asymmetric (SSL/TLS) encryption. The passphrase can be automatically generated or manually defined, as we will see later in the Gemini/Call-function ("Cryptographic Calling", which was introduced with GoldBug (and the Spot-on Kernel Project)), There are hardly any other - also open source - applications that include an end-to-end (e2e) encryption from one participant to

the other participant, in which the user can manually and individually define the passphrase (e.g. an AES string).

A (symmetric) end-to-end encryption is thus to differentiate from the point-to-point encryption. Therefore, the word “continuous” end-to-end encryption is also added (better still: continuous symmetric end-to-end encryption) - because it’s about that only the participant Alice and the participant Bob the secret passphrase know. Point-to-point encryption would be when Alice connects to the server and then the server connects to Bob. This may mean that the server can read the message, so it unpacks and repackages the message, especially if there is an asymmetric key between the participants and the server located in the middle.

Instead, GoldBug offers continuous symmetric end-to-end encryption that can not only be manually defined, but can also be instantaneously renewed with automation (this is called cryptographic calling, see below).

This special way of mixing PKI and AES as well as having a transfer via a SSL/TLS connection in place is referred to as echo protocol, which is to be deepened in the following section, because it still contains one further characteristics when sending to the network: a special feature when unpacking the encrypted capsule. So what exactly is the specific Echo protocol?

3 What is the Echo Protocol?

With the Echo-Protocol is meant - simply put - that

- Firstly, every message transmission is encrypted...

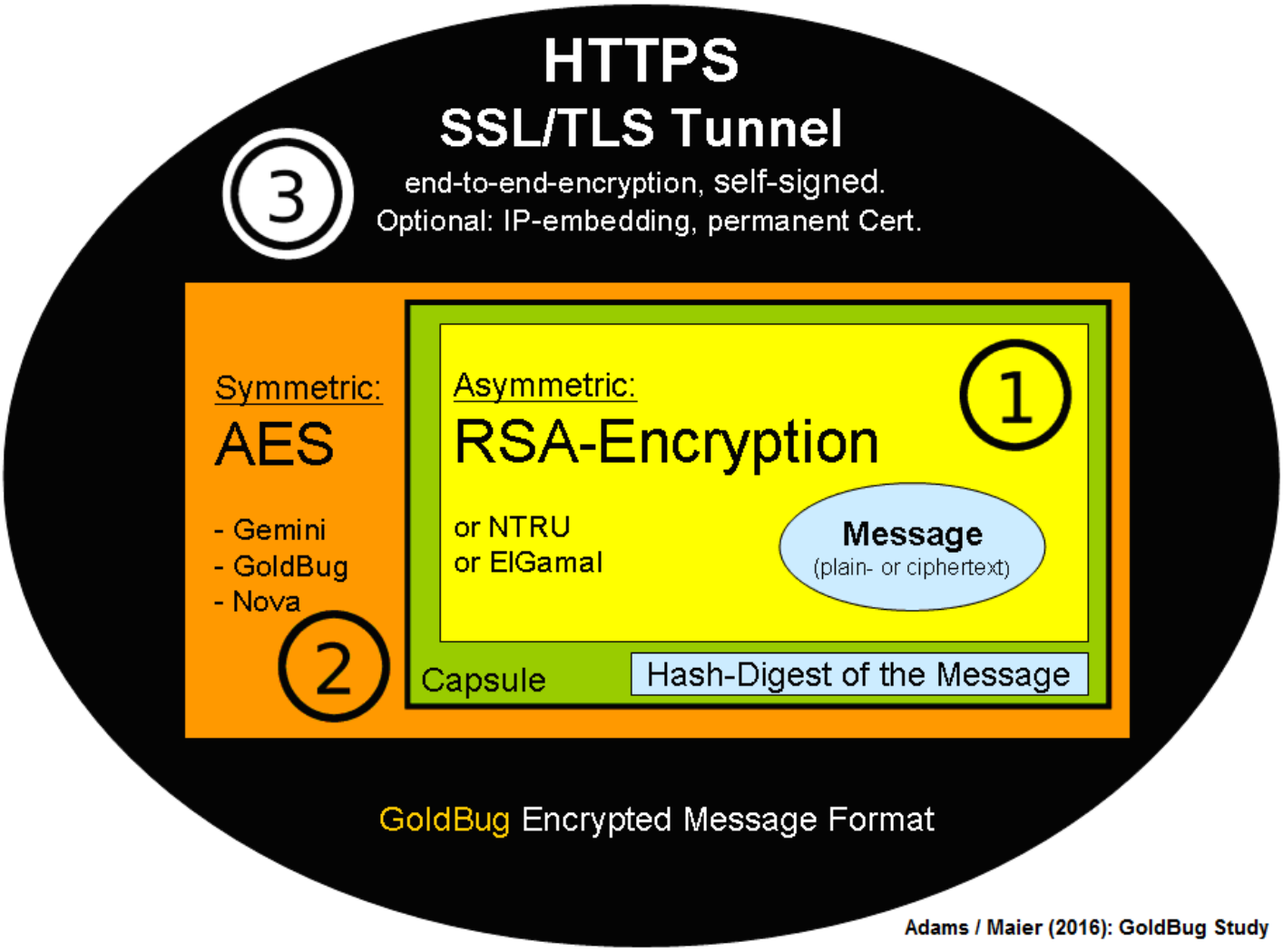
Example: SSL (AES (RSA * (message)))

*) instead of RSA you can also use Elgamal or NTRU or McEliece,

- ... and second, in the echo network, each connection node sends each message to each connected neighbor. Point. That’s how easy the world is. Underlying is the so-called “small-world phenomenon”: Anyone can somehow reach anyone over seven corners in a peer-to-peer or friend-to-friend network - or simply distribute the message over a shared echo-chat server in the circle of friends.

- A third criterion for the echo protocol can be added, that is a special feature when unpacking the encrypted capsule: The capsules have neither a receiver nor sender information included - and here they are different from TCP packets. The message is identified by the hash of the unencrypted message as to whether the message should be displayed and readable to the recipient in the UI or not. But for this so-called “echo match” see even more detailed below.

Figure: Format of the used echo protocol



The figure (according to Adams / Maier 2016) shows from inside to outside the process of how the encrypted capsule is formed in the context of the Echo protocol:

First level of encryption: The message is encrypted and the ciphertext of the message is hashed and then the asymmetric key (e.g. the RSA algorithm) can also be used to encrypt the symmetric keys. In an intermediate step, the encrypted text and the hash digest of the message are bundled into a capsule and packed together. It follows the paradigm: Encrypt-then-MAC. To prove to the recipient that the ciphertext has not been corrupted, the hash digest is first formed before the ciphertext is decrypted.

Third level of encryption: Then this capsule can be transmitted via a secure SSL/TLS connection to the communication partner.

Second level of encryption: Optionally, there is also the option of symmetrically encrypting the first-level capsule with an AES-256, which is comparable to a shared, 32-character password. Hybrid encryption is then added to multiple encryption (see Adams / Maier 2016:46).

The “half echo” mode sends a message only one hop, i.e. from Bob to Alice. Alice then stops sending the message (as is the default with the full echo).

Thirdly, in addition to Full Echo and Half Echo, there is the Adaptive Echo (AE). Here, the message is only sent to neighbors or friends, if they know an encryption token, they have

previously stored. So if the user does not know the token, the message will not be forwarded to this user.

After all, the echo still knows echo accounts. A kind of firewall. This ensures that only friends who know the account access can connect. So a web-of-trust can be created, which is a network exclusively among friends. It is not based on the encryption key but is independent of it. This means that the user does not have to associate his public key with his IP address or even announce it in the network.

Basically, in the echo, each node sends a message to each connected node: If a user should then receive a message a second time, it is compared in a temporary cache (based on the hash value for that message) and, if applicable, when the hash is known again, the message is discarded and thus not forwarded. This approach is called “congestion control” and balances the number of messages in the network from multiple nodes or servers.

A small analogy: The cryptography of the Echo Protocol can be compared with the giving and taking of so called “surprise eggs”, a capsule with a to assemble mini-toy in the famous chocolate egg. Bob gives Alice a surprise egg, Alice opens it and consumes the chocolate and bumps inside into the plastic capsule of the surprise egg, trying to open it and assemble the pieces into a toy, a smurf. However, she does not succeed in the assembly, the Smurf cannot be formed and therefore she packs the items back into the plastic capsule, pours new chocolate around and passes the egg to the neighbor, who also tries to assemble some of the pieces. Alice does not know who can assemble the surprise egg or the smurf successfully, so she continues to copy it (- what a miracle, Alice has an surprise-egg copying machine -) and gives each of her friends a copy. (Unpacking, crafting, evaluating, packing, giving away and unpacking, crafting, evaluating, wrapping, giving away, and so on ..

From the point of view of the entities represented in the network (kernels), the network would have become an surprise-egg paradise in this analogy, if the crafting processes were not reduced again with Congestion Control. Once known assembling parts are not built a second time together. Alice tinkers many packets until she recognizes a smurf with a red cap, she has received the figure of the Papa Smurf intended for her (or as her message).

In order to exclude time and frequency analyzes in the Internet or echo network, there are other functions in GoldBug which increase encryption or make crypto analysis more difficult:

For example: with the GoldBug application you can also send a kind of “fake” messages (from the simulacra function) and simulated communication messages (“impersonated messages”). On the one hand, encryption is not encryption, but it is pure random characters that are emitted from time to time, and the other is simulated human conversation, which is also based only on scrambled random characters:

Simulacra: This feature sends a “simulated” chat message to the echo network when the checkbox is activated. This “fake” message consists of pure random numbers, making it harder for analysts to distinguish encrypted messages with real and random messages. Simulacrum is

a term that is not unknown from both the movie “[Matrix]” (https://en.wikipedia.org/wiki/Matrix_(Film)) and Baudrillard’s philosophy (Neo uses this name for the repository for software in his home and the book “Simulacres et Simulation” by the French media philosopher Jean Baudrillard explores the relationship between reality, symbols and society). Several years after the publication of the Echo Protocol, similar donors to the Tor network have developed software called Matrix Dot Org, which sends encrypted capsules to the network similar to the Echo protocol and also addresses a messaging function; an analysis is pending where the echo over the plagiarism-like architecture offers differences and benefits or offered further open source suggestions.

Impersonator: In addition to random fake messages, the GoldBug program can also simulate a chat as if a real person chats from time to time and sends out replies. Also these messages are filled with pure random data, but they vary - simulated in a real chat conversation. Thus, analysis of messages can be made more difficult if third-party recorders should temporarily store and record all user communication, which may be assumed. But even more: even the absence of meta-data (see data retention) gives no reason to suspect that a message was for the user. Anyone who has been able to successfully unpack a message normally does not send it back to the echo network. A record of metadata could have increased interest in the un-re-submitted messages, assuming that this message could then have been successfully decoded by the user. For this case there is also the option of the SuperEcho:

SuperEcho: This feature also redirects successfully decoded and readable messages back to all friends. Failure to retransmit the message may then no longer indicate to the SuperEcho that the message may have been successfully decoded.

SuperEcho, Simulacra and Impersonation are three options of the GoldBug program, which should make it harder for attackers to understand the messages that are of interest to the user (and apparently others) in the multitude of messages.

Now let’s take a closer look at the individual echo modes:

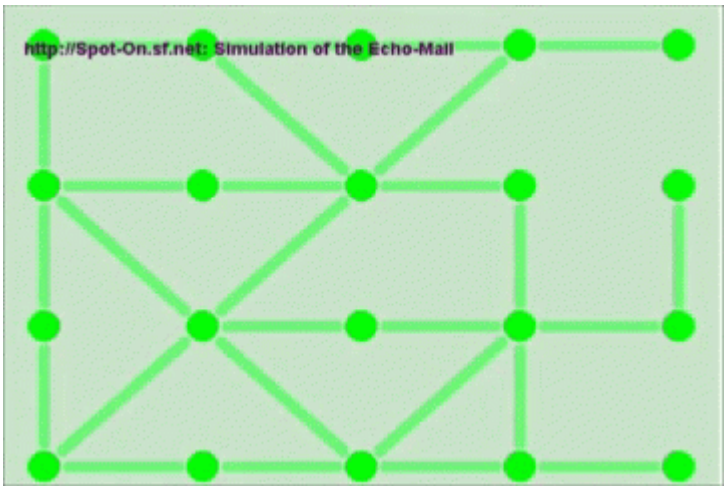
3.1 Full echo

The “full echo” modus underlies an assumption, as it is also in the so-called “small world phenomenon” given: hopping over a few friends everyone can send a message to each of them. Somehow, everyone knows everyone about a maximum of seven corners. This is also applicable in a peer-to-peer or friend-to-friend network. Therefore, a user can reach anyone if each node sends each message to all other known nodes (see figure).

Alternatively, a user can support this decentralized claim or abbreviate the message paths by installing an own chat server based on the Echo kernel for his friends, so that all encrypted messages can be sent to the participants and the server can serve as an e-mail-postbox or intermediate chat server.

The mapping simulates sending the message from a starting point to all network nodes across all connected network nodes.

Figure: Echo Simulation: Each node sends to each connected node



So basically, in the echo, each node sends each message to each node. This sounds simple: The echo protocol is a very simple protocol, but also has wider implications, that is: There are no routing information within the echo given and even metadata can hardly be recorded from the individual node or even network. The nodes also do not forward the message, the term “forwarding” is incorrect, because each node actively resends the message to the (its) connected friends.

This may result in receiving a message (from multiple connected nodes) multiple times - however, in order to avoid this happening and being efficient, the message hash is cached and the message may be rejected for retransmission if it is identified as a doublet. This is called as already indicated above: “Congestion Control”.

The message is in a capsule, so to speak, similar to a ZIP file. This capsule is created by asymmetric encryption with the public key. Added is the hash of the plaintext message. When a node tries to decode the ciphertext, a new text comes out - which can either be decoded correctly or incorrectly, that is to say, it is human-readable or, if the decoding key was incorrect, random characters became only random characters. This resulting text after the decoding attempt is thus again hashed.

Now, if the hash of the decoded message is identical to the hash of the original message that the sender already attached to the capsule, it is clear that the deciphering node has used the correct key and this message in plain text is for him: hence, the message is readable and displayed in the user interface. This can be called an “echo match”. Unsuccessful decoding attempts, in which the hash value between the original message and the message text of the decoding attempt do not match, are not displayed in the user interface, but remain in the kernel of the program for further transmission to the connected neighbors.

The node must therefore try with all the keys of his friends to unpack the message and compare the hash values. If the hash value is not identical, the node packs the ingredients back together

in one capsule and sends it to each of the connected friends, who then try the same.

The hash value of a message is not invertible, therefore the encryption cannot be broken with the (enclosed) hash of the original message, it still requires the correct key.

A message that has been successfully unpacked will no longer be sent, unless the user uses the “Super Echo” option, which also retransmits the successfully unpacked messages. Thus, no one who records the Internet packets can identify messages that are not sent again.

Finally, as described above, it is also possible from time to time to send out false messages (“simulacra fake messages”) and also simulated impersonated messages, so that it is difficult for network traffic collectors to find out the message capsule, which has been of interest for the users own readability. Because it is to be noted that it may be assumed today that all communication data of an Internet user is somewhere stored and recorded on the Internet and in the case of interest also automated and manually evaluated.

Then: This encrypted capsule is again sent over an encrypted SSL/TLS channel that is established between the nodes. This is a decentralized, self-signed p2p connection, a “two-pass mutual authentication protocol”. The implementation is precisely defined according to SSL/TLS, but it can also be switched off: The network nodes thus communicate via HTTPS or even HTTP.

However, of course, the transfer becomes more susceptible if one does not use the multiple encryption. Therefore, one should always establish an HTTPS connection to his friends and send over this encrypted channel his encrypted capsules in which the message waits, to be kissed awake from the right key (using the “echo match” method based on the hash comparison) and to be converted in readable plaintext.

|** Process description of the echo match:**| |—| |Sender A hashed his original text to a hash 123456789, encrypts the text and packs the crypto-text and hash of the original message into the capsule (before he adds an AES and sends it out via a TLS/SSL connection). Recipient 1 converts the received encoded text of the capsule to a (supposed) plain text, but this has the hash 987654321 and is therefore not identical to the supplied original text hash of 123456789. This is repeated with all available keys of all friends of the recipient 1, since all Hash comparisons, however, were unsuccessful, he re-packs the message again and sends it on. The message is obviously not for him or one of his friends. Recipient 2 now also converts the received, encrypted text to a (supposed) plaintext, this has the hash 123456789 and is thus identical to the supplied original text hash of 123456789, the decoding was apparently successful with one of the existing keys of his friends and therefore the message is displayed on the screen of this receiver (and if Super-Echo is selected, also re-packed again and sent-out again).|

No one on the net can see what message a user successfully unpacked, because everything happens on the user’s local machine.

3.2 Half echo

The half echo mode sends the user's message only one hop to the next node, e.g. from Bob to Alice. Alice then does not send the message down the path of her connected friends (as it is customary for the Full Echo). This Echo mode is technically defined by the connection to another listener: Bob's Node, when connecting to the node of Alice, notifies that Alice should stop sending the message to her friends. Thus, two friends or nodes can exclude via a direct connection that the message is carried into the wider network via the other, further connections that each node has.

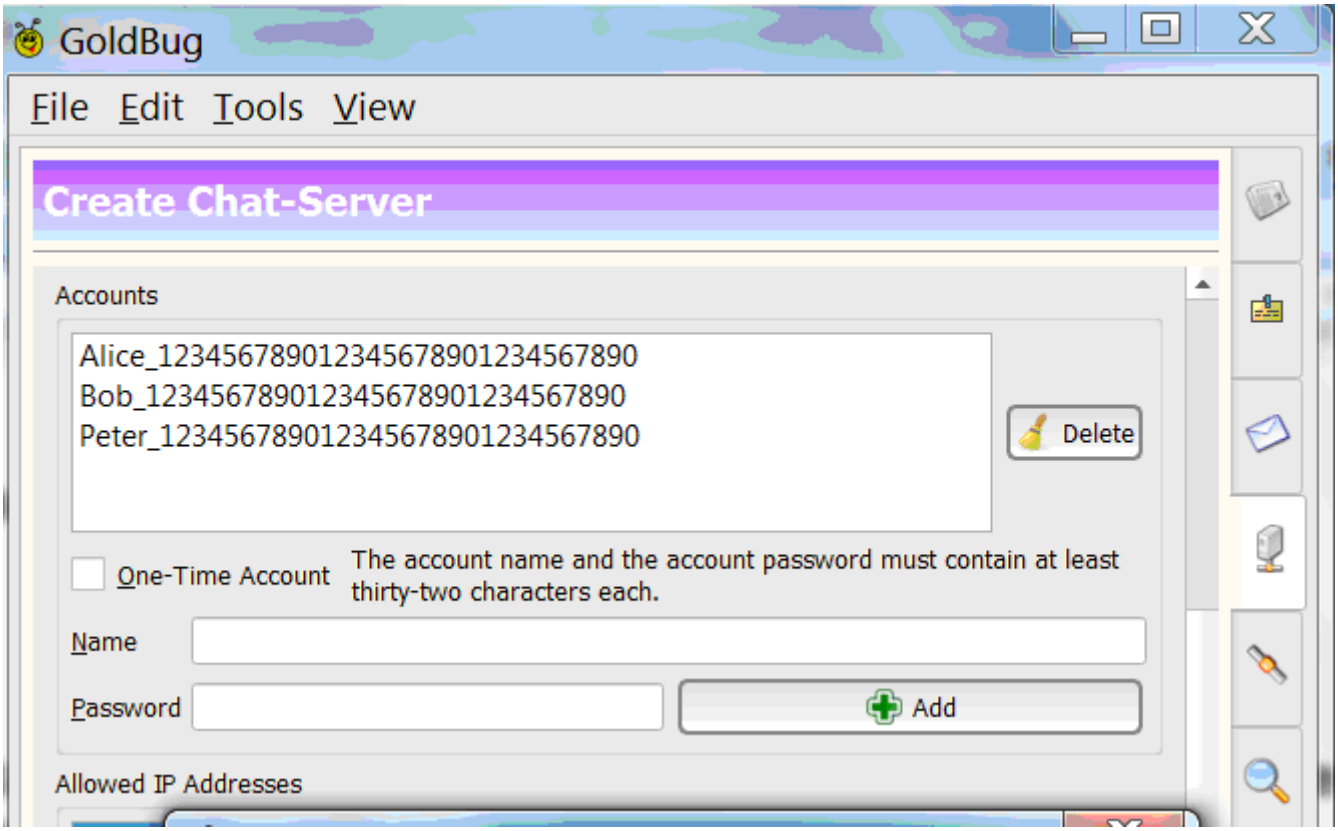
In addition to the Full and Half Echo, there is the third: Adaptive Echo (AE). Here, as it will be described below, the message is sent to connected neighbors or friends only if the node knows a particular cryptographic token - similar to a secret passphrase. Of course, this passphrase must first be defined, shared and stored in the respective nodes. Thus, defined ways of a message in a network configuration can be used. For example, if all nodes in a country use a common Adaptive Echo passphrase, the message will never appear in other nations' nodes if they do not know the passphrase. Thus, a routing can be defined that is not located within the message, but in the nodes. If you do not know the passphrase, one does not get the message forwarded respective further sent out! Adaptive Echo turns messages that cannot be opened into messages that are not known or even exist.

The section below on the Adaptive Echo (AE) will therefore cover this option in more detail.

3.3 Echo Accounts

And in addition: the echo also knows Echo Accounts. An account is a kind of firewall. It can be used to ensure that only friends connect who know the credentials to the account. Thus, a so-called Web of Trust, a network based on trust, is formed. It is not based on the encryption key like in other applications, it is independent of it. This has the advantage that the encryption public key does not need to be associated with the IP address (as it is the case with RetroShare, for example); or that the user must announce his IP address in the network of friends, for example in a DHT where users can search for it. The echo accounts provide a peer-to-peer-(P2P)-connection to a Friend-to-Friend-(F2F)-network or allow both types of connection. This makes GoldBug suitable for both paradigms.

Figure: Figure: Account Firewall



The echo accounts work as follows:

Binding endpoints are responsible for defining the account information. During the creation process for an account, this can be defined for one-time use (one-time account or one-time use). Account name and also the passphrase for the account require at least 32 bytes of characters. So a long password is required.

After a network connection has been established, the binding endpoint informs the requesting node with a request for authentication. The binding endpoint will drop the connection if the peer has not identified within a fifteen second time window.

After the request for authentication has been received, the peer responds to the binding endpoint. The peer then transmits the following information: HHash Key (Salt / Time) // Salt, where the hash key is a concise summary of the account name and also the account password.

Currently, the SHA-512 hash algorithm is used to generate this hash result. The time variable has a resolution of a few minutes. The peer retains the value for the cryptographic salt.

The binding endpoint receives the information of the peer. Consequently, this then processes HHash Key (Salt // Time) for all accounts he has set up. If the endpoint cannot identify an account, it will wait one minute and perform another search. If an account matching this hash key was found, the binding endpoint creates a message similar to the one the peer created in the previous step and sends the information to the peer. The authenticated information is stored. After a period of about 120 seconds, the information is deleted again.

The peer receives the information of the binding endpoint and performs a similar validation process, this time including the analysis of the cryptographic salt value of the binding endpoint.

The two salt values must then be clearly consistent. The peer will drop the connection if the endpoint has not identified itself within a fifteen-second time window.

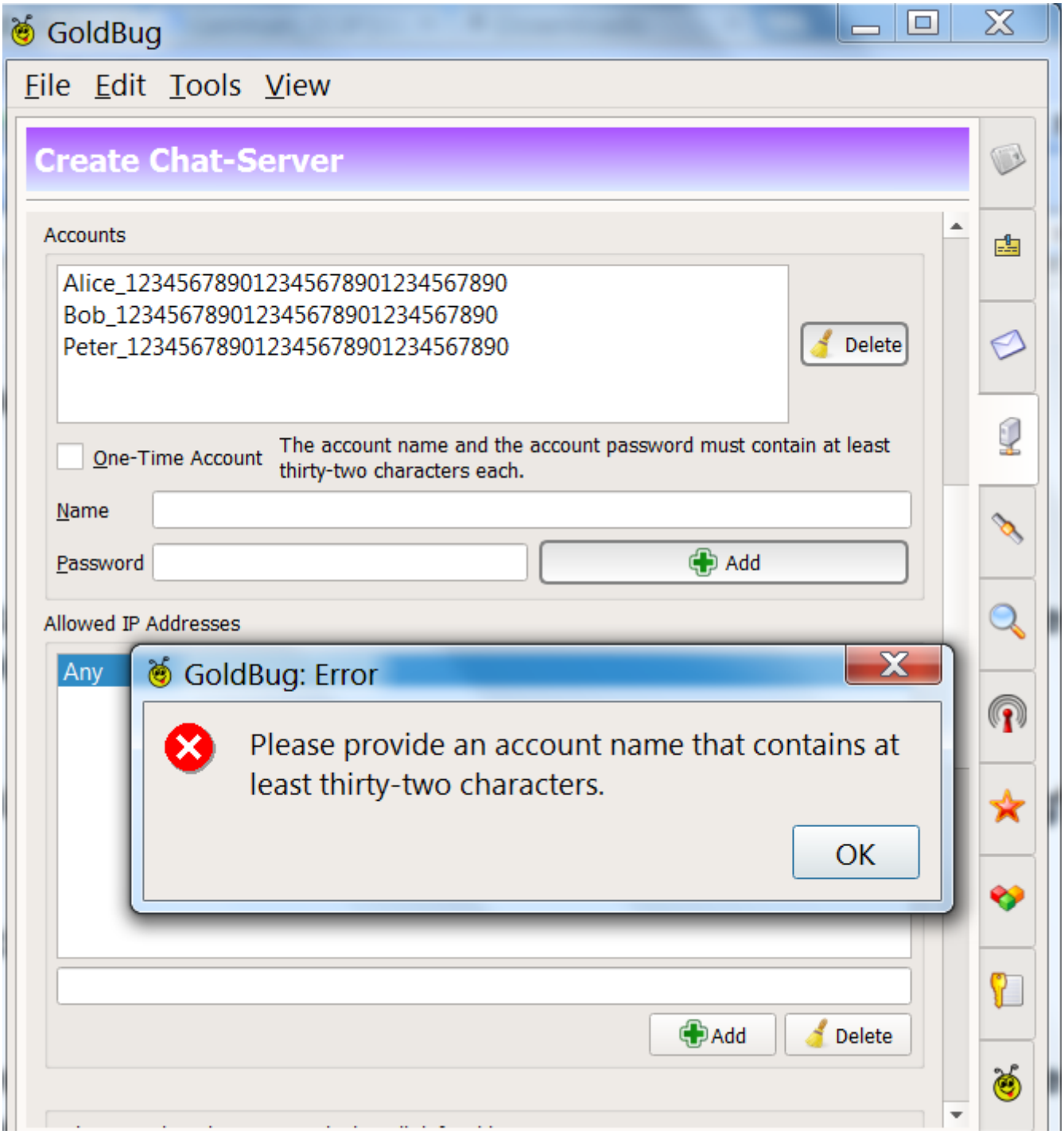
It should be noted, by the way, that the account system can be further developed by including a key for encryption. The additional key then allows even more precise time windows to be defined.

If SSL/TLS is not available during this negotiation, the protocol may become vulnerable as follows: An intermediate station may record the values from the third step and consequently send to the binding endpoint.

Then, the binding endpoint could also grant access to the account to an unknown connection. The recording device could then grab the response of the binding endpoint, that is, the values of the fourth step, and forward the information to the peer. If the account information or password is then accurately maintained, the peer would then accept the response from this new binding endpoint. That’s why, as always, it’s about protecting passwords.

In GoldBug, therefore, a server account - if it is specified to be dedicated - therefore requires a password equal to the length of an AES-256: this is a passphrase of 32 characters.

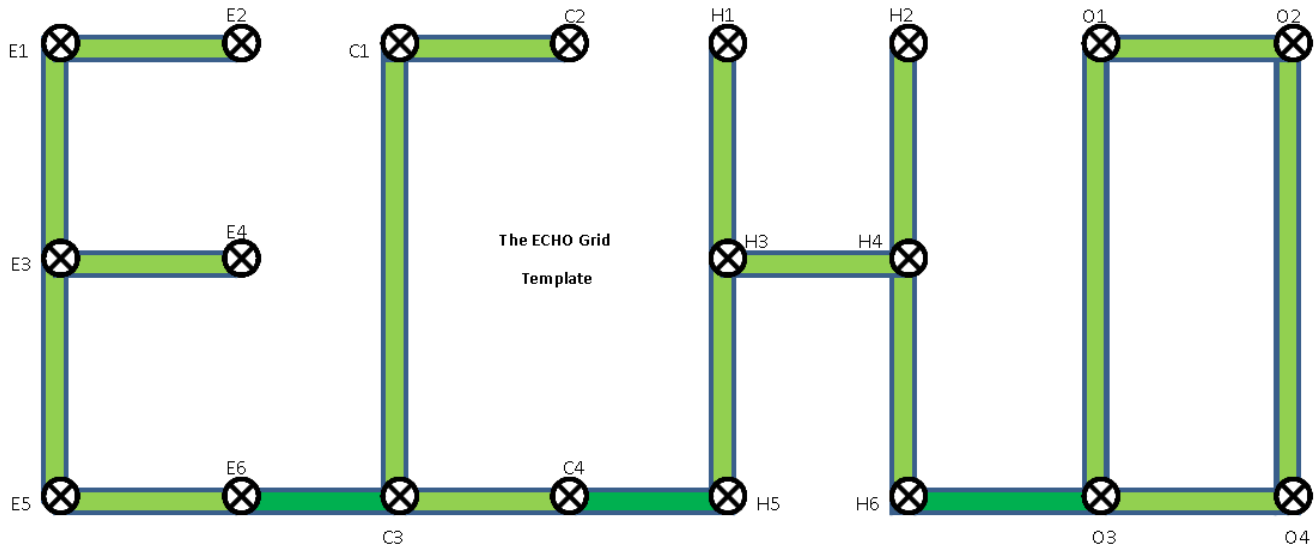
Figure: Account Passwords



3.4 The echo grid

When students talk or students be taught or taught themselves about the Echo Protocol, they can simply draw an echo grid with the letters E_C_H_O. The nodes from E1 to O4 are numbered and connect the letters with a connecting line on the ground (see figure).

Figure: The Echo Grid Template



For example, then the connection E1-E2 denotes an IP connection to a neighbor.

If the individual nodes now exchange keys, connections are created that arise as a new level at the level of the IP connections of the P2P / F2F network.

With the architecture underlying in GoldBug not only the cryptographic routing in a kernel program was elaborated, also - as stated above - the term “cryptographic routing” was paradoxically removed from the routing with the echo protocol. It is therefore necessary to speak in more detail of the “cryptographic echo” instead of “cryptographic routing”. One more of them to differentiate the protocols, is the protocol of the “Cryptographic Discovery” which will be discussed below in an extra section.

Echo is thus “beyond” routing: Firstly, the message packets do not contain routing information (addressees) and the nodes also do not use “forwarding” in the original sense, because they simply send everything to all connections. And secondly: Even the cryptographic key that tries to decode the message is not an address (which would even be attached to the message package), but only a polarizing glass: it lets us see texts differently and possibly understand. The echo protocol therefore also uses the term “traveling” rather than the term “routing”. Or just in short: “cryptographic echo”.

From a legal point of view, a different evaluation is then also to be made here, since a node does not forward in the name of an addressee as a middleman, but informs the neighbors independently (see, for example, the forwarding in other routing models such as AntsP2P with its ant algorithm, Mute, AllianceP2P, RetroShare, Onion-Routing or I2P.

As well as spreading a good reputation in the neighborhood, the message also spreads in the echo - otherwise the echoing protocol allows any cryptographic “stuff” to “float” away (by not being decoded or being unreadable).

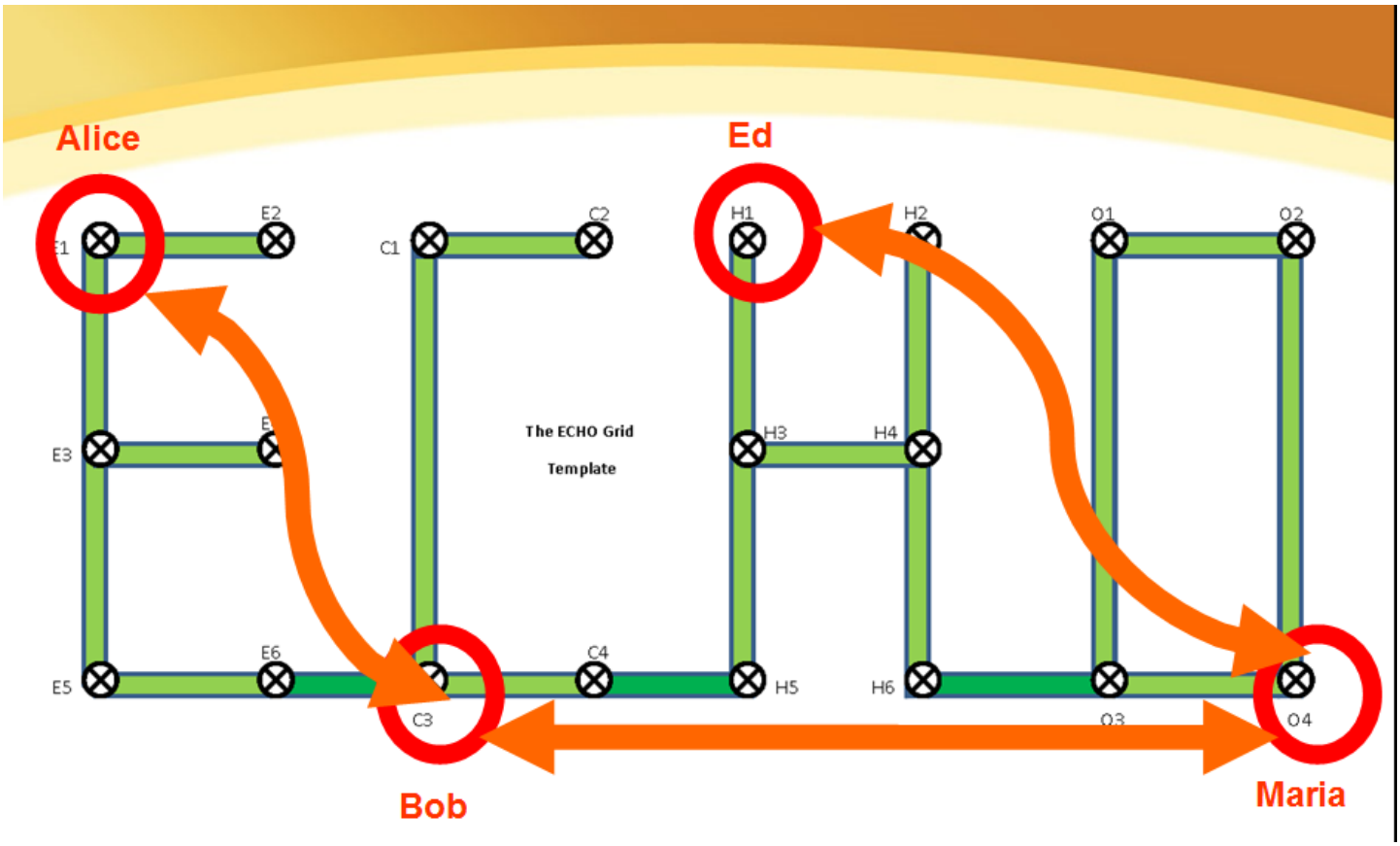
It is reminiscent of the Star Trek Borg collective paradigm: everyone has access to all the neighbors’ messages (unless half or adaptive echo is used and if the message text can be understood (decoded) at all).

In the echo, the node is more of a “sovereign” or “giving and receiving (non-directional) information”; in other networks, a node could be more referred to as a “postman”, “dealer”, “forwarder” or “intermediary”.

The echo grid as a simple network representation is not only used for the analysis of “routing” (or “travel”-ways) to represent echo modes and encryption stations, but ultimately can also be found in graph theory application: which path takes a message, depending on the structure of the network, and also can be evaluated the use of echo accounts, half or full echo and the Adaptive Echo, as the following examples of the graphs between Alice, Bob, Ed and Maria illustrate.

3.4.1 Examples of key exchanges by Alice, Bob, Ed & Maria

Figure: Alice, Bob, Ed and Mary in the Echo Grid - An example of the echo



The following examples of the figure can be further discussed - they use a few vocabulary and processes of functions of the GoldBug client, so that in the program inexperienced readers can also skip this section and refer back once the basic functions (installation, chat, e-mail, File transfer or URL search) have been explained – so that these technical examples are understood at a later date):

Alice (IP = E1) and Bob (IP = C3) have exchanged their public key and are connected via the following IP neighbors: E1-E3-E5-E6-C3.

Bob (C3) and Maria (O4) are also friends, they've also swapped their public keys for encryption: and use their neighbors' IP connections: C3-C4-H5-H3-H4-H6-O3-O4.

Finally: Maria (O4) is a friend of Ed (H1). They either communicate via the path: O4-O3-H6-H4-H3-H1 or they use the path of: O4-O2-O1-O3-H6-H4-H3-H1. Since, in the echo protocol, every IP neighbor sends every message to every connected IP neighbor, the path that delivers the message fastest will succeed. (The second incoming message is then filtered out by Congestion Control).

Direct IP connections from neighbors such as E1-E3 can be further secured by the creation of a so-called "echo account": No other IP address than E1 can then connect to the so-called "listener" of the neighbor E3. This method can be used to create a web-of-trust - without relying on keys for encryption - nor does it require a friend as a neighbor to exchange their chat or e-mail key.

So-called "turtle hopping " becomes much more efficient in the echo network: when Ed and Alice start a file transfer (via the StarBeam function and using a magnetic URI link), the echo protocol transports the packets via the path H1-H3- H5-C4-C3-E6-E5-E3-E1. Maria is not in the route, but she will also receive the packets over the full echo if she knows the StarBeam magnet. The advantage is that the hopping is not done over the keys, but over the IP connections (e.g. the Web of Trust). Basically, everything is always encrypted, so why not take the shortest route on the net?

A so-called "Buzz" or "echo'ed IRC Channel" (therefore also short: e'IRC) space can be created or "hosted" by the nodes O2, for example. Since only the user Ed knows the buzz room name, all other neighbors and friends are left out. Advantage: The user can talk to unknown friends in a room without having exchanged a public e.g. RSA key with them. Instead, a one-time magnet is simply used for this "buzz" / "e'IRC" room.

Maria is a mutual friend of Ed and Bob and she activates the C/O (care of) feature for emails: this allows Ed to send emails to Bob, even though he's offline, because: Maria saves the e-mails in her instance, until Bob comes online.

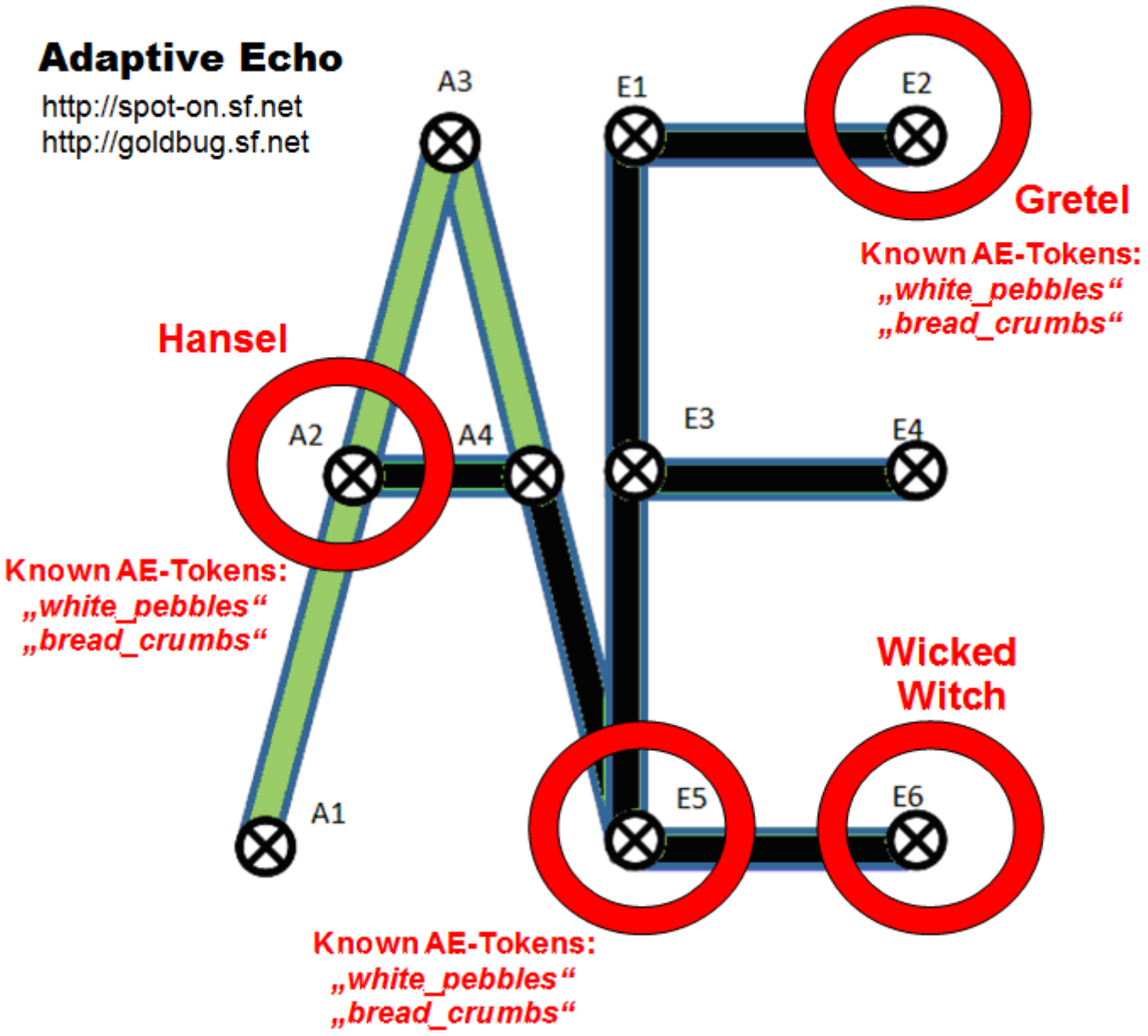
Furthermore: Alice created a so-called virtual "e-mail institution". This is not comparable to a POP3 or IMAP server because the emails are only cached: Ed sends his public email key to Alice - and Ed adds the magnet of Alice's "E-mail Institution" to the program. Now, emails from Bob and Ed are also cached within Alice (at the email institution), even though Maria should be offline.

It is helpful to follow the examples on the above graphic or to come back to this at the end of the manual after further explanations of the functions.

3.5 Adaptive Echo (AE) and its AE tokens

For the explanation of the "adaptive echo" another echo-grid with the connected letters A and E can be drawn (see following figure).

Figure: Adaptive Echo (AE): The “Hansel and Gretel” Example of the Adaptive Echo



If a user, his or her chat friend, and a configured third node as a chat server insert the same AE token (“Adaptive Echo Token”) into the program, then the chat server will only send the user’s message to his friend, and not to all other connected neighbors (or users), as it would normally be the case in the Full Echo mode.

The AE token consists, like a passphrase, of at least 96 characters. In the case of Adaptive Echo, the information from the sending node of the encrypted capsule is attached - and all other nodes learn that you are only forwarding (sending) the message to nodes or connection partners, who also know this AE token.

With an AE token, no other node that does not know the passphrase will be able to receive or view the user’s message. Thus, potential “recorders” can be excluded, these are possible neighbors, which presumptive record all the message traffic and then try to break the multiple encryption to get to the message core of the capsule.

In order to be able to determine the graph, the travel route, for the adaptive echo, several nodes must agree with each other and note the passphrase on the way path without any gaps. In the case of the adaptive echo it can be spoken of a routing.

3.5.1 Hansel and Gretel - An example of the Adaptive Echo mode

To illustrate the adaptive echo, a classic example is the tale of Hansel and Gretel.

In the AE grid explained above, the characters Hansel, Gretel and the evil witch are drawn as nodes. Now Hansel and Gretel think about how they can communicate with each other without the evil witch noticing this. According to the fairy tale, they are in the woods with the witch and want to find out again of this forest and mark the way with bread crumbs and white pebbles.

These fairy tale contents can now also illustrate the Adaptive Echo in the above grid pattern and show at which points of the grid or the communication graph a cryptographic token called “white pebbles” can be used:

If nodes A2, E5 and E2 use the same AE token, then node E6 will not receive a message that node A2 (Hansel) and node E2 (Gretel) will exchange. Because the node E5 learns about the known token “white pebbles”, it does not send the messages to the node E6, the “Wicked Witch”. It is a learning, adaptive network.

An “adaptive echo” network reveals no target information (compare again above: “Ants Routing”). Because - as a reminder: The mode of “half echo” sends only one hop to the connected neighbor and the “full echo” sends the encrypted message to all connected nodes over an unspecified hop number. While “echo accounts” encourage or hinder other users as a kind of firewall or authorization concept when connecting, “AE tokens” provides graph or path exclusivity - for messages that are sent via connected nodes that know the AE token,

Chat Server Administrators can exchange their tokens with other server administrators if they trust each other (so-called “Ultra-Peering for Trust”) and define a Web of Trust. In network labs or at home with three or four computers, the Adaptive Echo is easy to test and to document the results:

For an adaptive echo test, simply use a network with three or more computers (or “SPOTON_HOME” as the (endless) file in the binary directory to launch and connect multiple program instances on a single machine) and then implement this example flow:

- Create a node as a chat server.
- Create two nodes as a client.
- Connect the two clients to the chat server.
- Exchange keys between the clients.
- Test the normal communication skills of both clients.
- Put an AE token on the server.
- Test the normal communication skills of both clients.

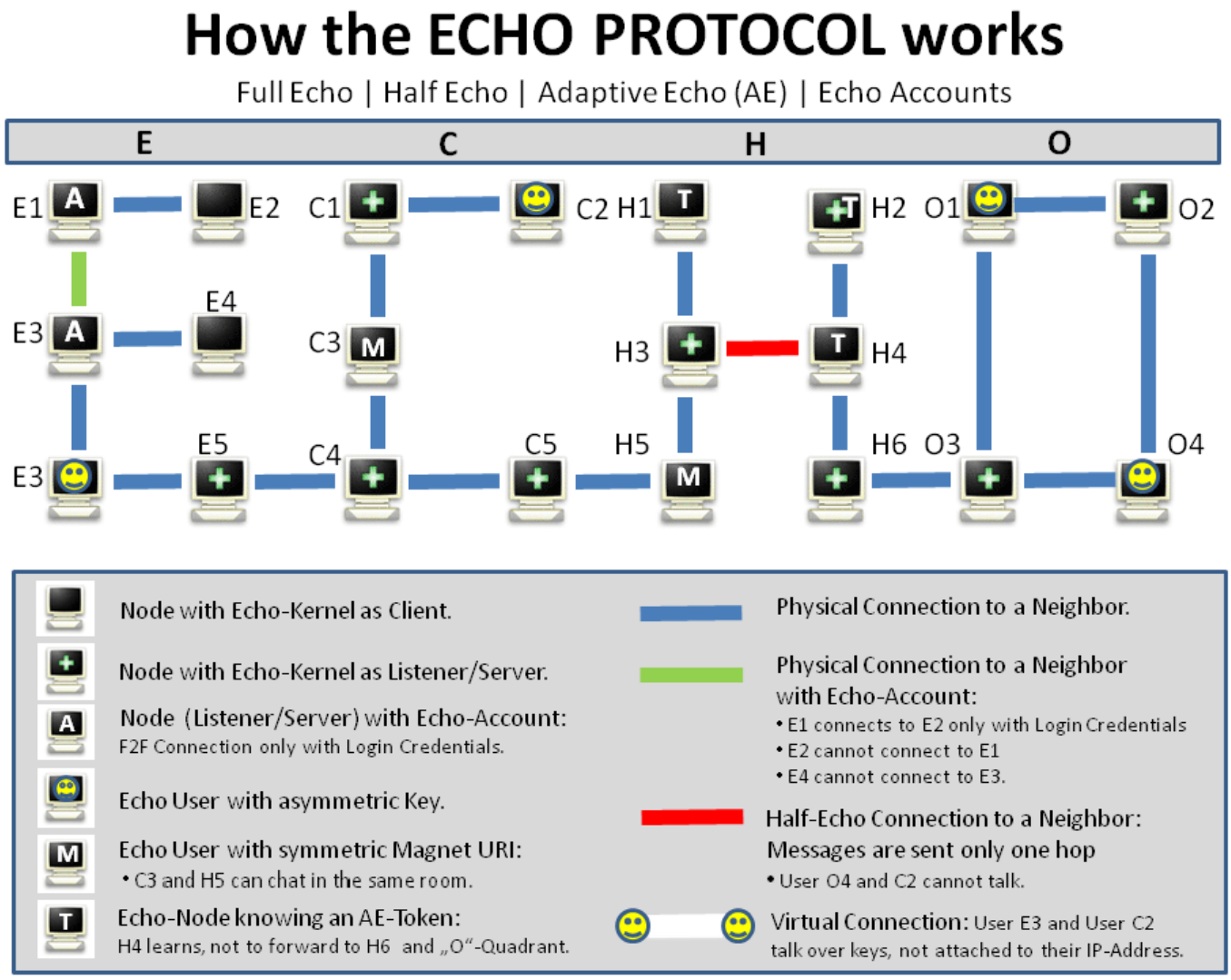
- Now set the same AE token in a client as well.
- Note the result: The server node no longer sends the message to other nodes that do not have or know the AE token.

This example should be easy to replicate.

3.6 Some examples of how the echo protocol works

Taking the various methods and options together, the figure “How the echo log works” will provide a more complex overview.

Illustration: How the echo PROTOCOL works



- Shown in the graphics are the different usage examples of “Full Echo”, “Half Echo”, “Adaptive Echo” and “Echo Accounts”.
- A distinction is made between physical IP connections and virtual connections to keys. Keys are therefore not necessarily assigned to an IP connection.

- Users can exchange asymmetric public keys as well as magnet URIs with symmetric encryption details as well as tokens and account credentials.
- Connected nodes can allow and deny connections - as well as send messages addressed (or sent addressed).
- Accordingly, different communication scenarios arise.

Examples:

- a. User H4 has an AE token. It does not send messages (via connection node H6) to the O-quadrant if HG does not know the token.
- b. If H3 sends a message to H4, then H4 will not send this message because it is a “half echo” connection.
- c. The user E1 does not let the user connect E2 because he does not know the login for the echo account.
- d. User O1 and O4 chat with each other and only know their public key for encryption.
- e. User H3 and C5 chat via a URI magnet in the same group chat room (also called buzz or e’IRC).

It turns out that the echo protocol can map a very complex network, although it has a simple structure. With the individual options and terms, users can tap into a lot of network and crypto theory in practice and try it out in practice. An ideal teaching tool, introduction to cryptographic processes, graph theory, and a hands-on user experience in the group. For example, for a team of three, as in the GoldBug story by Edgar Alan Poe.

4 Cryptographic Discovery

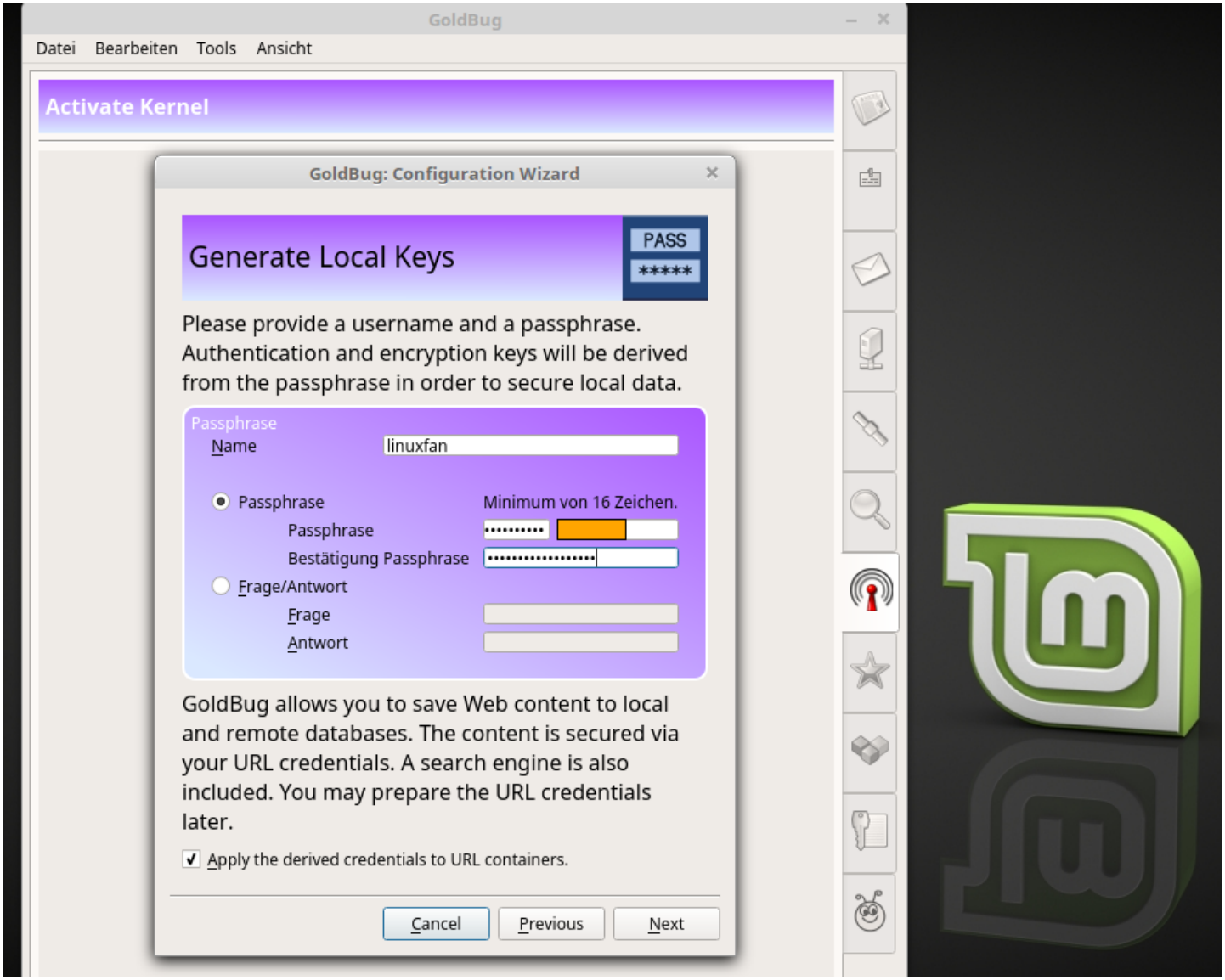
Cryptographic Discovery describes the method of an echoing protocol to find nodes in an echo network. The echoing protocol is supplemented with another useful method, if not even more important, than the echo itself. Cryptographic Discovery is available in existing clients such as GoldBug, Spot-on or the chat server for the Android operating system, SmokeStack, implemented within the code base. The source code and its documentation define the method accordingly. For example, Cryptographic Discovery can replace a Distributed Hash Table (DHT) to find a friend on the network. A further publication especially on this topic as an attachment to the source code of the development is in preparation.

5 Set up a first installation – e.g. with the wizard

The first initial setup of the software is very simple in a few steps,

- The user unpacks the program from the Zip and starts (under Windows) the GoldBug.exe from the path to which the program was unpacked, e.g. C: /GoldBug/GoldBug.exe or C: /Programs/GoldBug/GoldBug.exe.
- The user interface and a wizard appear, with which settings can be implemented step by step. Alternatively, the user can close the wizard and make the settings manually in the tab for the settings or for kernel activation. It is recommended to use the wizard.
- In the wizard, the necessary cryptographic keys are then generated with the user name and a passphrase to be entered twice.

Figure: Initial Wizard



- After completing the wizard, the kernel must still be activated. So the GoldBug Messenger has a user interface (also called Interface or Graphical User Interface (GUI)) and a kernel. Both are given as a binary file (in Windows called GoldBug.exe and Spot-On-Kernel.exe). Spot-On is the original project for the Echo application and GoldBug merely provides a simplified user interface.

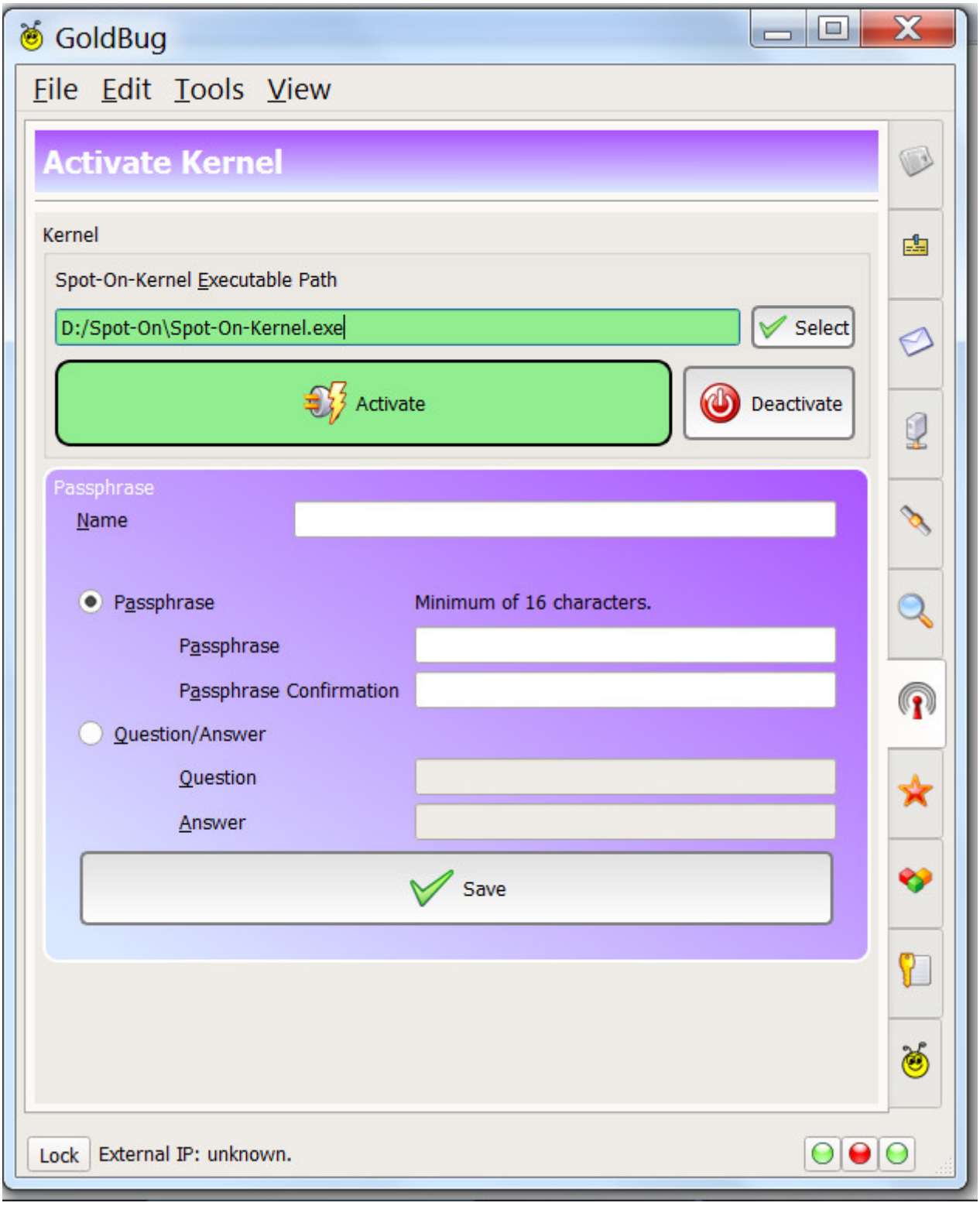
- If the kernel is running, the user connects to a neighbor or server with the appropriate IP in the Neighbors tab.
- Then the user exchanges the key with a friend and the encrypted communication via chat or e-mail can begin...

Otherwise, the user must activate the kernel via the “Activate” button in this tab for settings or for kernel activation after each start of GoldBug.exe, which then coordinates the connections to friends or to a chat server. So the kernel file Spot-on-Kernel.exe will be turned on or off from GoldBug’s program.

5.1 Two login methods

If the user starts GoldBug for the first time, the user enters a nickname in the corresponding box and defines a passphrase for the login into the application (see figure, blue widget box).

Figure: Set password



There are two methods to do this: the passphrase method or the question-and-answer method.

The password must be at least 16 characters long. If this is too long, you can repeat a shorter password three times, such as “password_password_password”, but then the password is not as secure as one with a random string.

The two methods can be differentiated as follows:

Passphrase method: When the password is created, it is not stored locally, just the hash of the input. The hash is supplemented by a supplementary string, the so-called cryptological salt. This complements the hash and makes it safer. The “Salted Hash” is thus defined as follows: hash

(passphrase + salt). To achieve that the password is also trained for the user and typing errors are excluded, it must be entered a second time.

Question / Answer Method: This method does not enter a password twice but defines a string as a question and a string as the answer. Both strings will not be checked a second time. Technically, this login method is implemented via an HMAC: Hash (Question, Answer), indicates that an “HMAC” (Keyed-Hash Message Authentication Code) is used. And: neither the question nor the answer is stored on the user’s machine and no cryptographic salt is randomly generated by the machine. Instead of the question, the user can of course also enter two passwords without a question mark. It should be noted that here the question and the answer must be entered in subsequent logins exactly as they were defined and here at the first definition no second input check (“confirmation”) regarding typing errors is given as in the password method.

Figure: Login to the application with a password

Since the hash generated from the login passphrase unlocks the encrypted containers that also store the private key for encryption, it is especially important to protect the login process and login password. Therefore, the above two methods have been taken into account to make it more difficult for attackers: they do not know a) which method a user has chosen and b) the question-answer method is safer, as described above, because neither the question, nor the answer can be stored somewhere and a HMAC may be more complex than a password as a "just" salted hash. Only the user knows question and additionally the answer and only the match of both can open the container.

In order not to reveal information to keypad loggers, there is the possibility to use a virtual keyboard when logging in (see image). The user starts this by double-clicking on the input line for the password. At best, only mouse clicks can be recorded here, but no keystrokes.

Figure: Virtual Keyboard

In principle, it is important that the private key is kept encrypted in a sufficiently secured container. It is reasonable to suppose that, in particular, access by providers to mobile operating systems would otherwise make it easy to fetch the private key.

This is especially critical for web mail offers to provide the encryption in the browser or with keys that are deposited at and with the mail provider online. Encryption should always take place on the user's machine and for this purpose, an open-source client and no online web application in the browser should be used, in which the user should - if necessary - also generate self-generated keys online.

The risk of seizing the possibly insufficiently encrypted private key is far too great. Program audits should also pay attention to capturing passwords for the encrypted container in which the private key is located, as well as to remote accessing the private key.

Even the few open-source messengers with encryption that can be counted on one hand for the desktop as well as for mobile devices that have undergone a security audit are hardly sufficient with regard to the security of the encrypted storage of - and secured access processes to - analyzed private keys.

5.2 Generation of 10 Keys for Encryption

When the user launches the GoldBug Messenger for the first time, the wizard asks if the user wants to generate the encryption keys. For key creation the user should choose a key of at least 3072 bits (default) or larger. The user can also choose other options such as algorithm, hashtype, cipher, salt-length or iteration count, for example, if he re-generates the key. The first setup has a presetting based on RSA ready: So if you want to test out NTUR or McEliece as an algorithm, then after the first setup again generate new keys with one of the then selectable algorithms.

The generated keys are stored in the sub-path "/.spot-on". If the user wants to set up a new login with new keys and all user data should be deleted, then this path can simply be deleted and the GoldBug.exe has to be restarted. For Linux and the other operating systems their adequate path specifications apply accordingly. The same can be done in the main menu with "!!! Total_Database Erase !!! ".

Asymmetric keys are generated for the following functions (a key for the encryption as well as a key for the (optional) signature):

- Chat: This is about the 1: 1 chat
- E-Mail: this is about email to other users of GoldBug or Spot-on
- POPTASTIC: This is about the chat via e-mail server
- URLs: This involves searching for URLs in the URL database (web search)
- Rosetta: With the Rosetta encryption pad, text with asymmetric keys can be converted back and forth from plaintext to ciphertext and vice versa before the texts are sent. This is recommended when other insecure messengers or e-mail are used or the ciphertext should be posted anywhere on the web - or the plain text, before it is sent in GoldBug, should again receive an additional encryption level.

That each function uses its own key pair is again a security feature. If the chat key were compromised, the email encryption will not be affected. Furthermore, the user can only pass friends his chat key and not the e-mail key. Thus, the user can decide whom he allows to chat with or just to e-mail or possibly also to exchange URLs for the function of p2p web search in the integrated URL database.

So far the minimal view on the user interface is described: Via the main menu one can choose between "full view" or "minimal view". If you are not familiar with computers, you should choose the minimal view because it fades out the unnecessary variety of options. Keep it simple! Qt developers, and those who are looking for an exercise project for their own Qt development project, may even minimize the user interface (and are invited to "fork" the GoldBug project).

During the first setup, the option of the maximum view is not available, it will only be shown and set in the other logins. The possibility of looking at even more details in the user interface should therefore be addressed briefly here, since many details also refer to the last-mentioned point of the cryptographic values, which is also contained in the tabulator of the kernel activation and encryption key: Key-Generation is there to be found (just in the setting of the maximum view).

The values can be set individually for a new key generation (without wizard) from the extended user view. However, if you are using the client for the first time, the typical setting values are automatically available, i.e. the key has a (predefined) size of 3072 bits of the RSA algorithm.

In case of a non-minimal view, for example, the tab “Activate Kernel” shows the following elements in the user interface:

Path to kernel: Here the user can enter the path to the kernel. If the kernel with the “spot-on-kernel.exe” in the path specified correctly, then the path is highlighted in green. Otherwise, you have to look at the executable file of the kernel or copy it to the executable file of the GUI (GoldBug.exe) or adjust the path accordingly.

PID: The PID number identifies the process ID that identifies the executable file in Windows. The user also finds the process IDs in the Windows Task Manager.

“Key regeneration” function: With the “re-generation” function, the user can also generate individual keys - with new values and options. For this the check box has to be activated, the values have to be set and the respective keys have to be re-generated. This is the way to get e.g. keys of the McEliece or NTRU algorithm. Then the user has to put his new key back to his friends, because the key is the communication ID.

Another variety of options can also be found under the main menu / options in a pop-up window, which will be explained later. This is the actual options window, which can be neglected for a first start, however.

Figure: Options e.g. for digital signatures

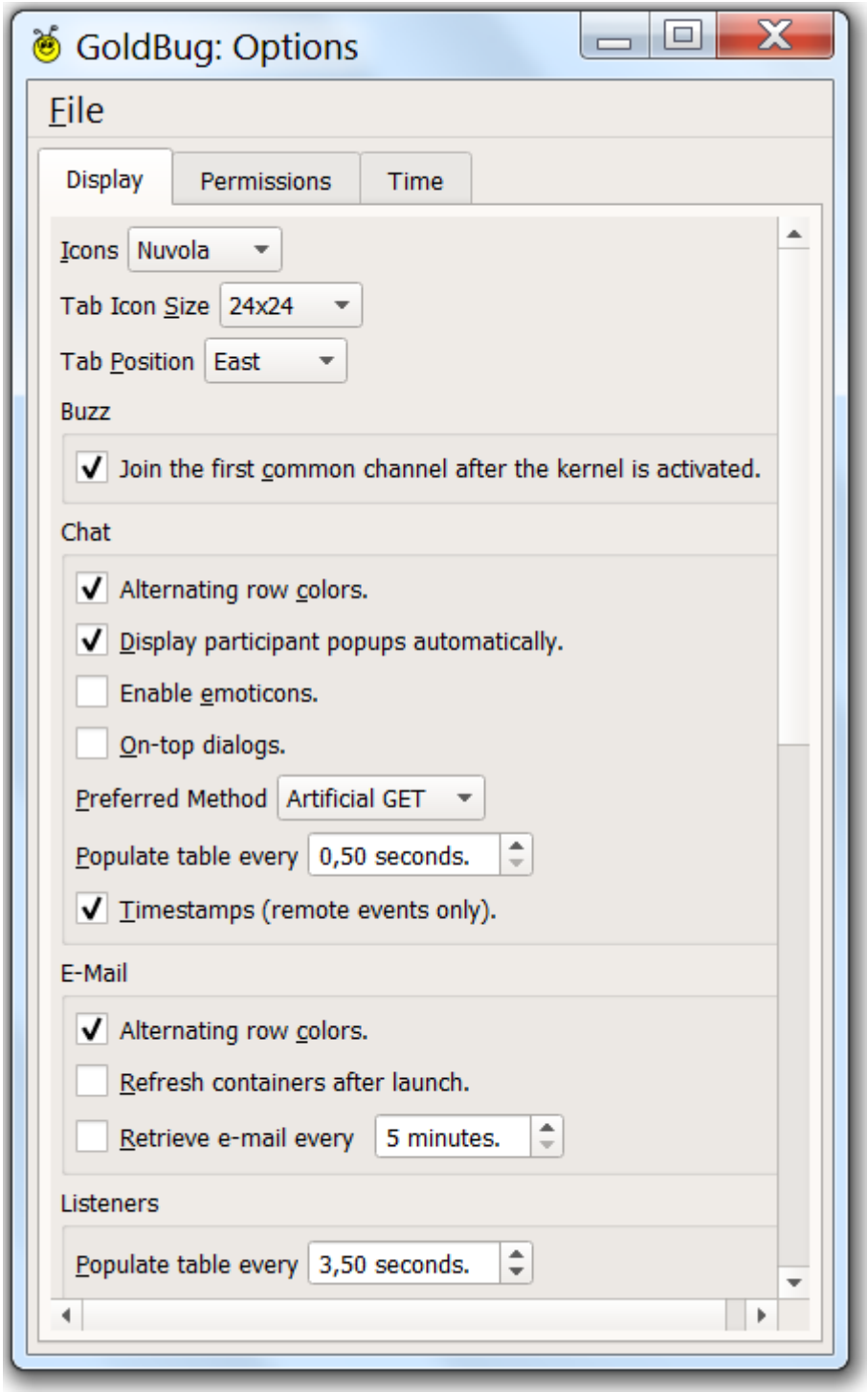
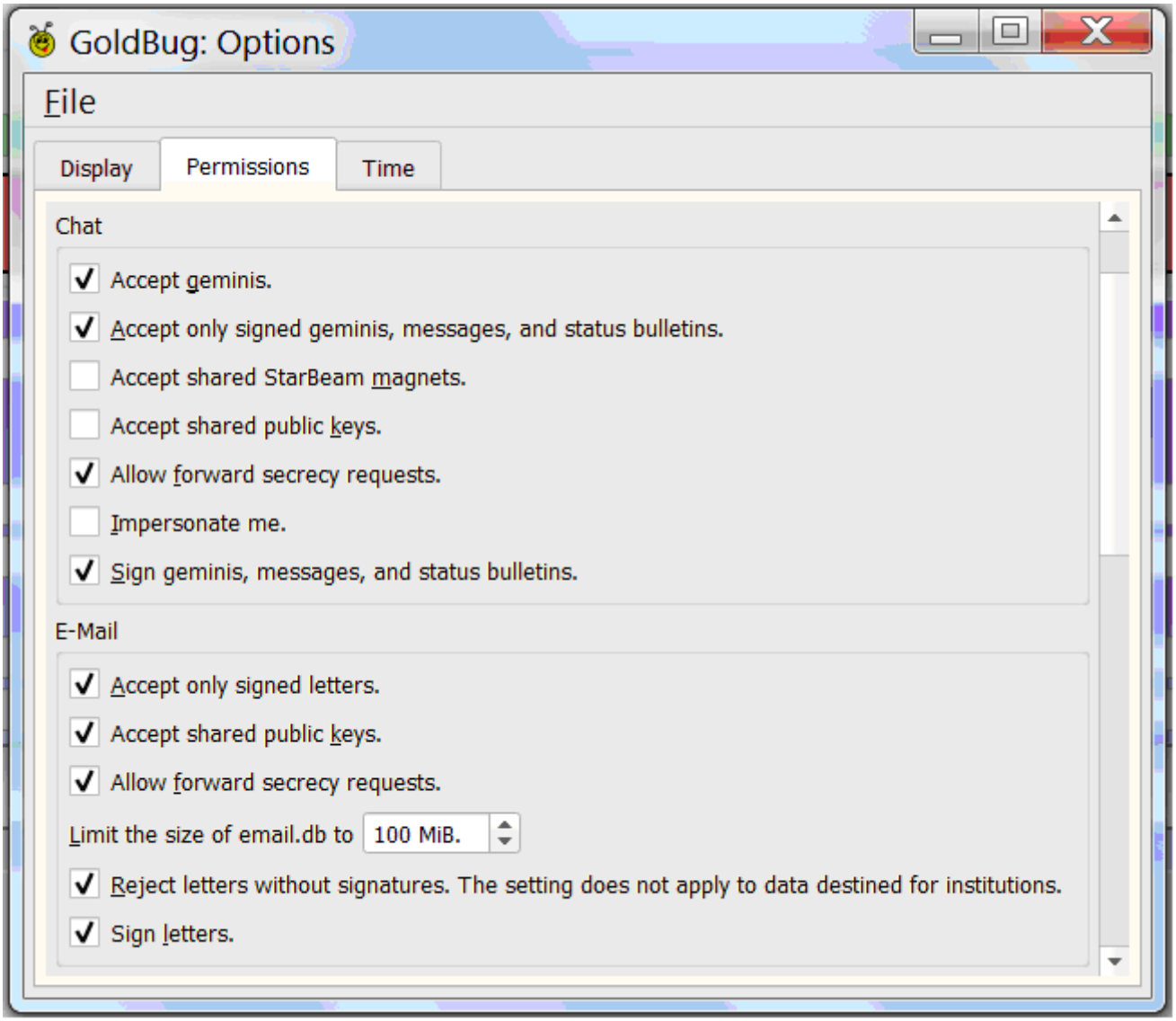


Figure: Options e.g. for the view in the client



It is more important to start now the kernel after the first key generation via the wizard.

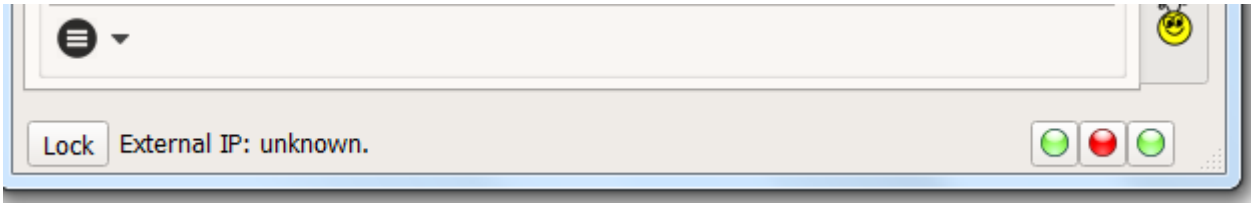
5.3 Activation of the kernel

When the user launches the GoldBug Messenger for the first time, a pop-up window at the end of the wizard asks if the kernel should be activated. Otherwise, the red “Activate Kernel” button should be pressed on all subsequent starts after the login in the settings tab, then the user can start: If the button is green, the kernel is running.

When the user closes the program interface, a pop-up window also asks if the kernel should continue running. So it’s a good idea to first deactivate the kernel and then close the GUI of GoldBug if you want to completely close the program.

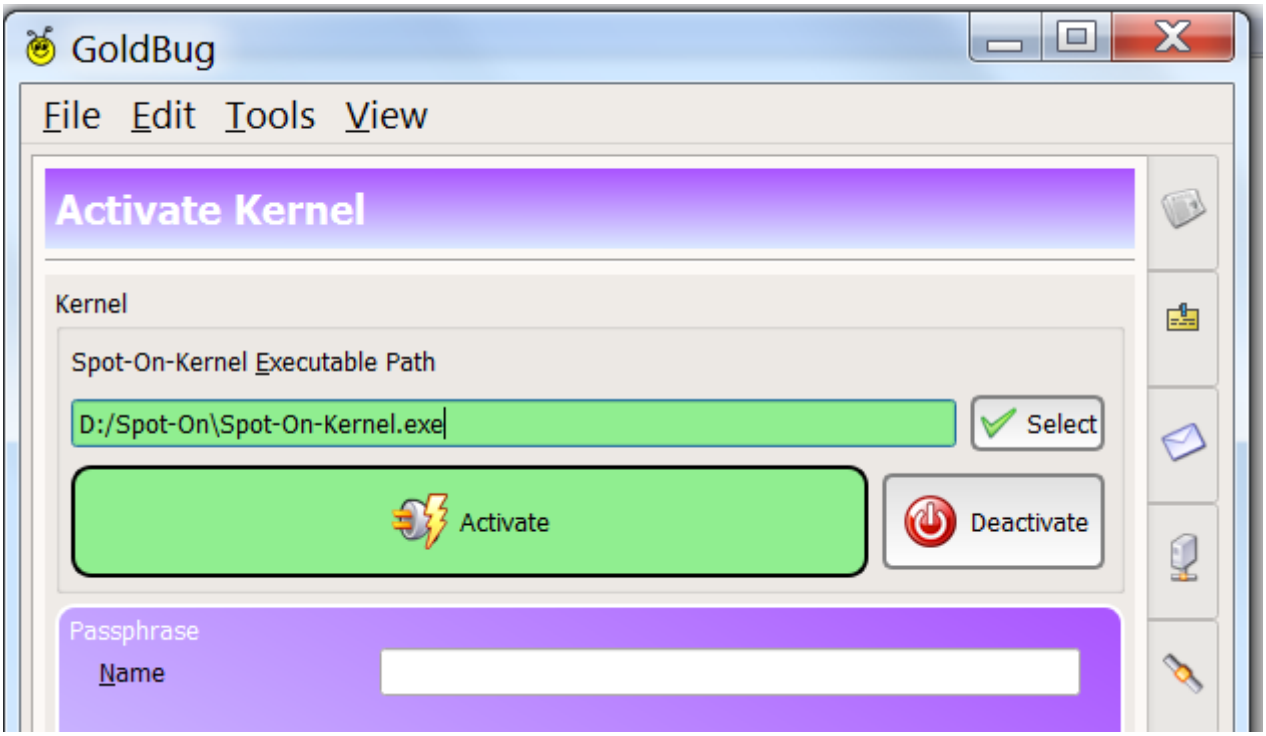
Otherwise, the user runs the kernel without a GUI, which is sometimes desired on a web server, so that nobody can manipulate within the open user interface. (In addition to the Spot-on kernel, there is also the Spot-on-Lite kernel for this daemon web server purpose, which can be found in the repository of the source code as a standalone repository.)

Figure: Lock of the user interface in the status bar



If the user wants to leave the GUI in place, but no one should be able to enter or change anything during the absence, it is also possible to click the “Lock” button on the left in the lower status line, the user interface will close and return to the login tab for the input of the password back, so that the running processes and inputs of other tabs are not visible. To unlock the interface, press the lock button again in the status bar and the user then enters the passphrase(s) in a pop-up window.

Figure: Activation of the kernel



The user can also enable/disable the kernel by pressing the first LED in the status line at the bottom left. If it is green, the kernel is active; if it is red, the kernel is off. The middle LED indicates whether the user has set up a listener / chat server and the third LED indicates whether the user has an active and successful connection to a neighbor / server.

Figure: Encryption between kernel and GUI / UI

The connection of the user interface (goldbug.exe) to the kernel (spot-on-kernel.exe) is also encrypted, although both run on the same machine. A tooltip on the mouse over the first LED indicates the encryption.

5.4 Connect a neighbor with the IP address

Upon initial activation, the IP address of the Project Chat Server is automatically added as a neighbor. This serves as a temporary chat server through which the user can chat with his friends test-wise until a separate connection node has been created on a web server or at home (or two users connect directly to each other). The test server will not last forever, so far, users will need to first set up a server themselves before they can connect two clients.

Up to now, the user has been connected to a chat server directly after activation of the kernel by the previous test server. If the user would like to add another, the tab “Connect neighbor” must be used. Here is an input field for the IP address of the neighbor respective the web server, on which a Spot-On Kernel is running or a friend also uses a GoldBug Messenger.

Figure: Add a neighbor / server

Figure: Connection to a neighbor server

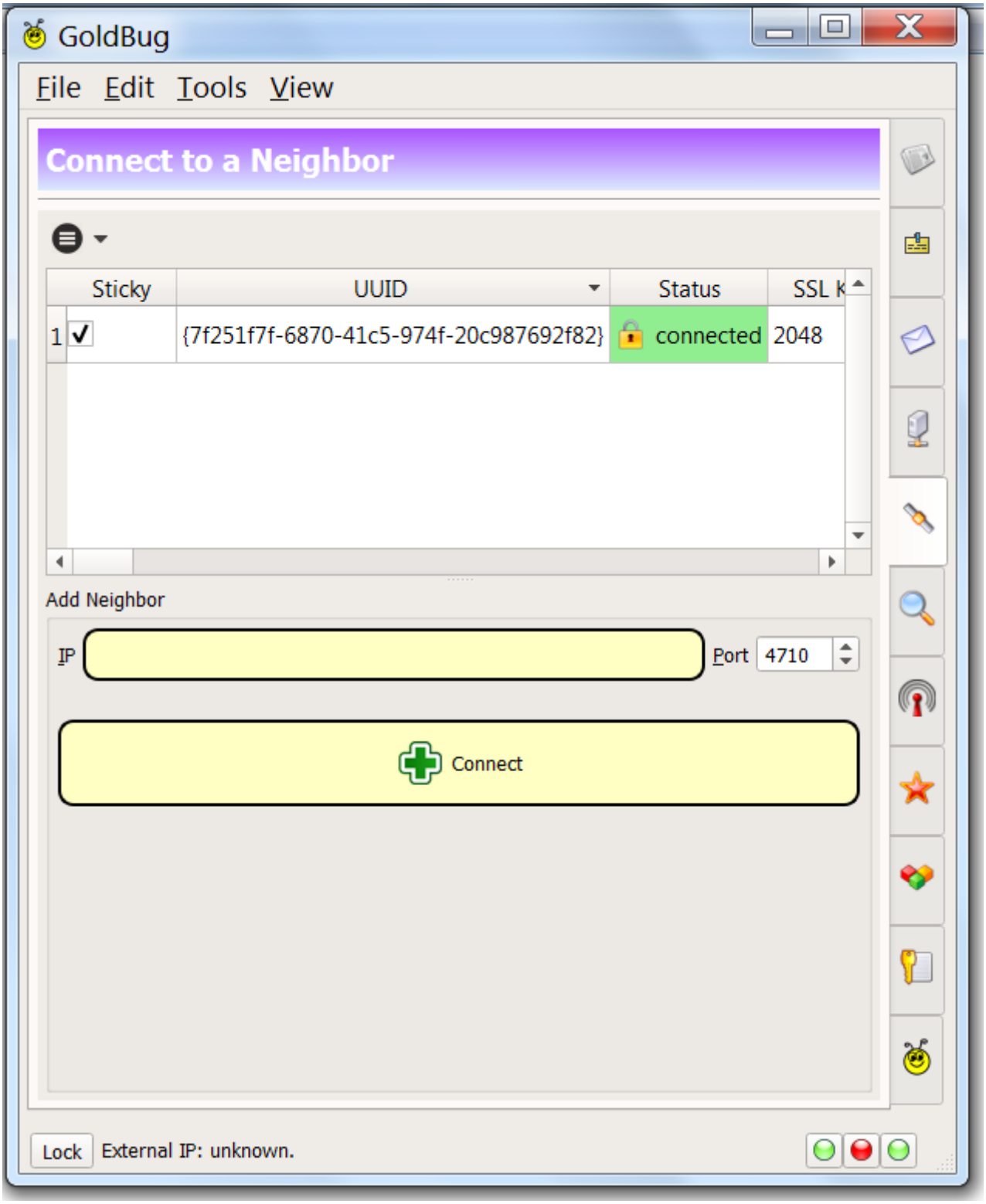
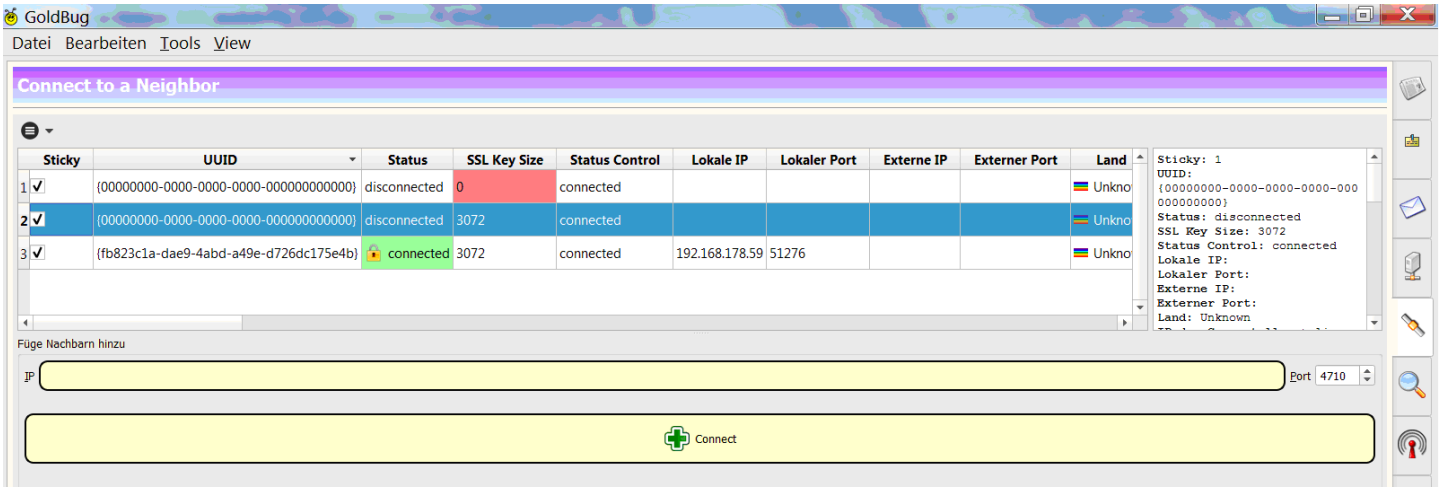


Figure: Connecting neighbor server



In the field, enter the IP address of the neighbor node. The points are each separated by three digits of the IP address (according to IP-V4). If a block only contains two digits, e.g. 37.100.100.100, then the 37 can be placed arbitrarily in the first block or entered as 37 in the first two positions. Then press the “Connect” or “Add” button. The IP address is then stored on the default port 4710 and appears as a link in the neighbors table.

If an error message appears, then this IP address has already been entered. In order to delete all neighbors, the button “Delete all neighbors” can be pressed (via the context menu button or via the right mouse button in the table in which the neighbor appears) and the IP address can be entered again. Optionally, you can also delete the file “neighbors.db” in the installation path “./spot-on” on the hard disk. It reforms immediately and is then empty.

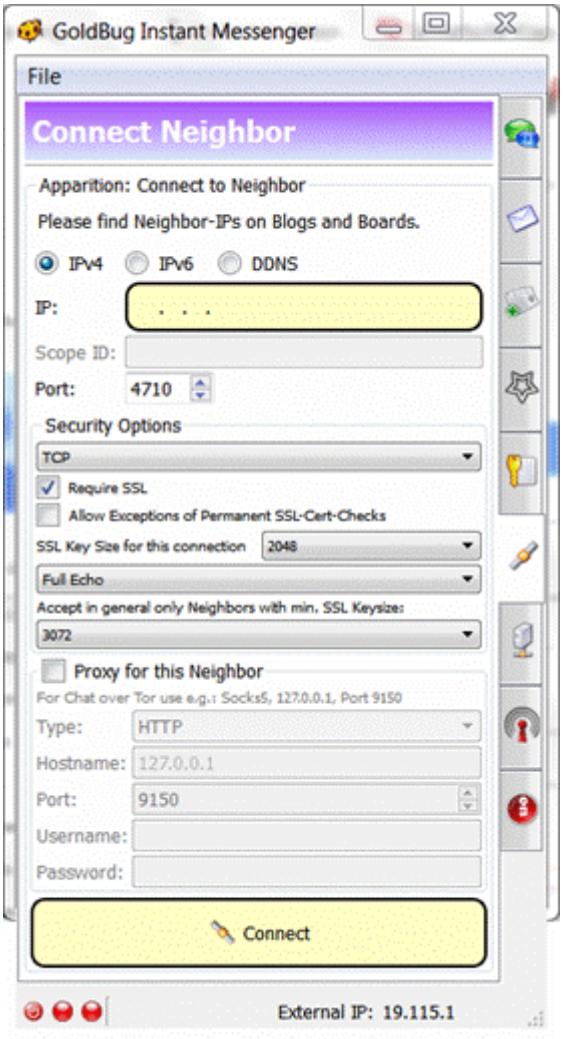
When the kernel is activated (left, first LED in the status bar is green) and the neighbor or server is connected (middle LED is green) everything is successfully installed and online. Entering an IP address and pressing the connect button should be quite easy.

If the user wants to connect directly to another user without a server, one of them must create a so-called listener in the tab chat server (and release the firewall for the port and, if necessary, forward the port in the router to his machine, see below in more detail).

Or: if both users are on the same Windows network, the existing neighbor “239 ..” can be activated, then the GoldBug Messenger is converted into a Lan messenger and finds all other GoldBug participants in the local LAN automatically and connects them as a neighbor. If the users then exchange the keys, the communication can start.

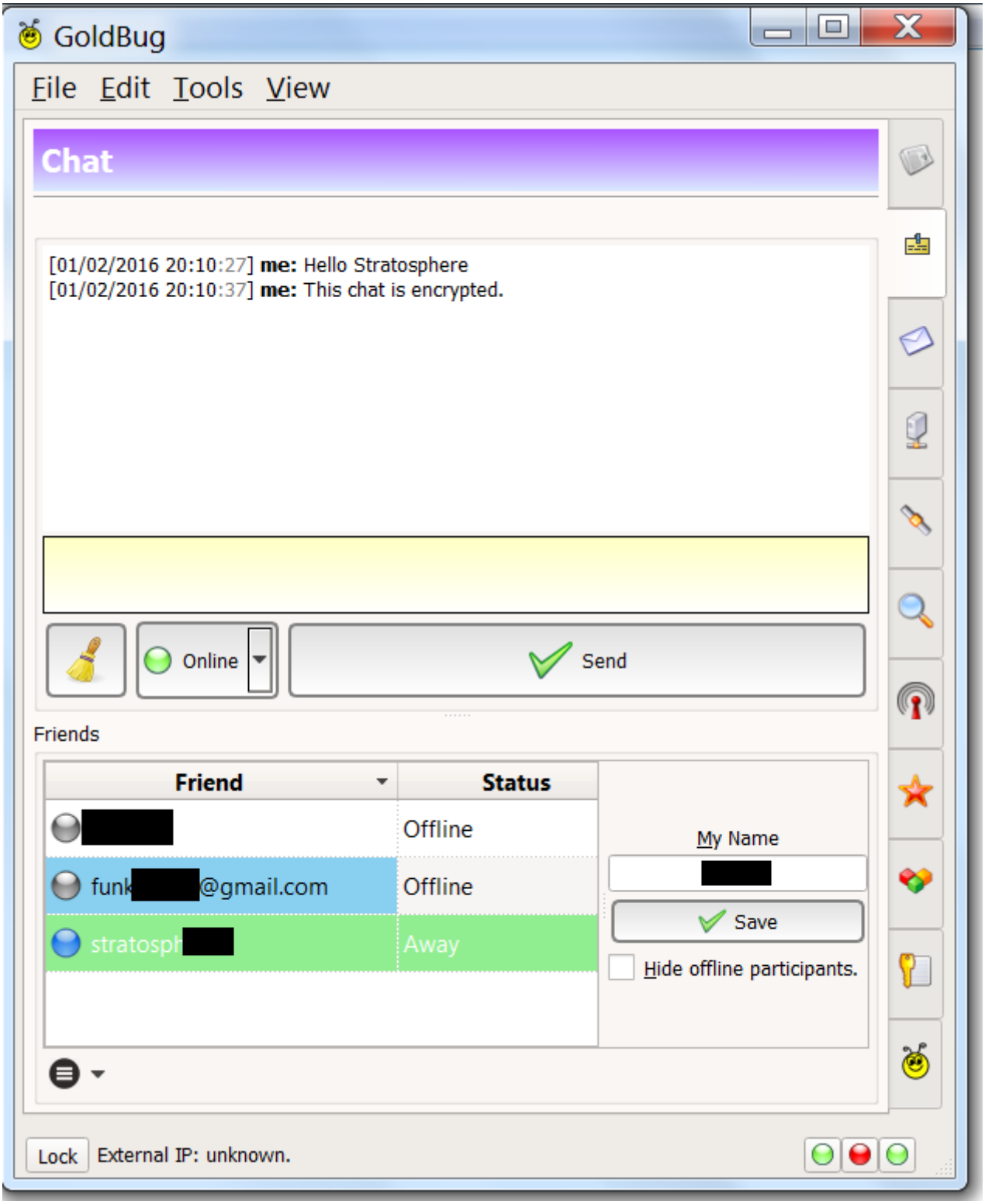
If you want to see more details, the minimal view can also change to the full view: In this view, it becomes clear that in addition to the IP address, the port of the IP address can also be individually configured. By default, GoldBug uses the port 4710. Furthermore, the program can also be operated via IPv6 as well as to a listener/server, which is linked via the dynamic DNS. Then DNS is no number sequence for the IP, but a domain name to be added in a textfield. Further security options can be defined in the box below or the connection server can also be addressed via a proxy (e.g. if the user wants to use GoldBug via the TOR network).

Figure: Add a neighbor with IP address (detailed view)



6 The chat function

Figure: Chat Tab



Now if login password is defined, key generated, kernel enabled and a neighbor-server connected, so in the status bar two LED lights are green, then the user can exchange his key with a friend and the communication can start in the chat tab (see figure) or in the pop-up windows for a defined participant.

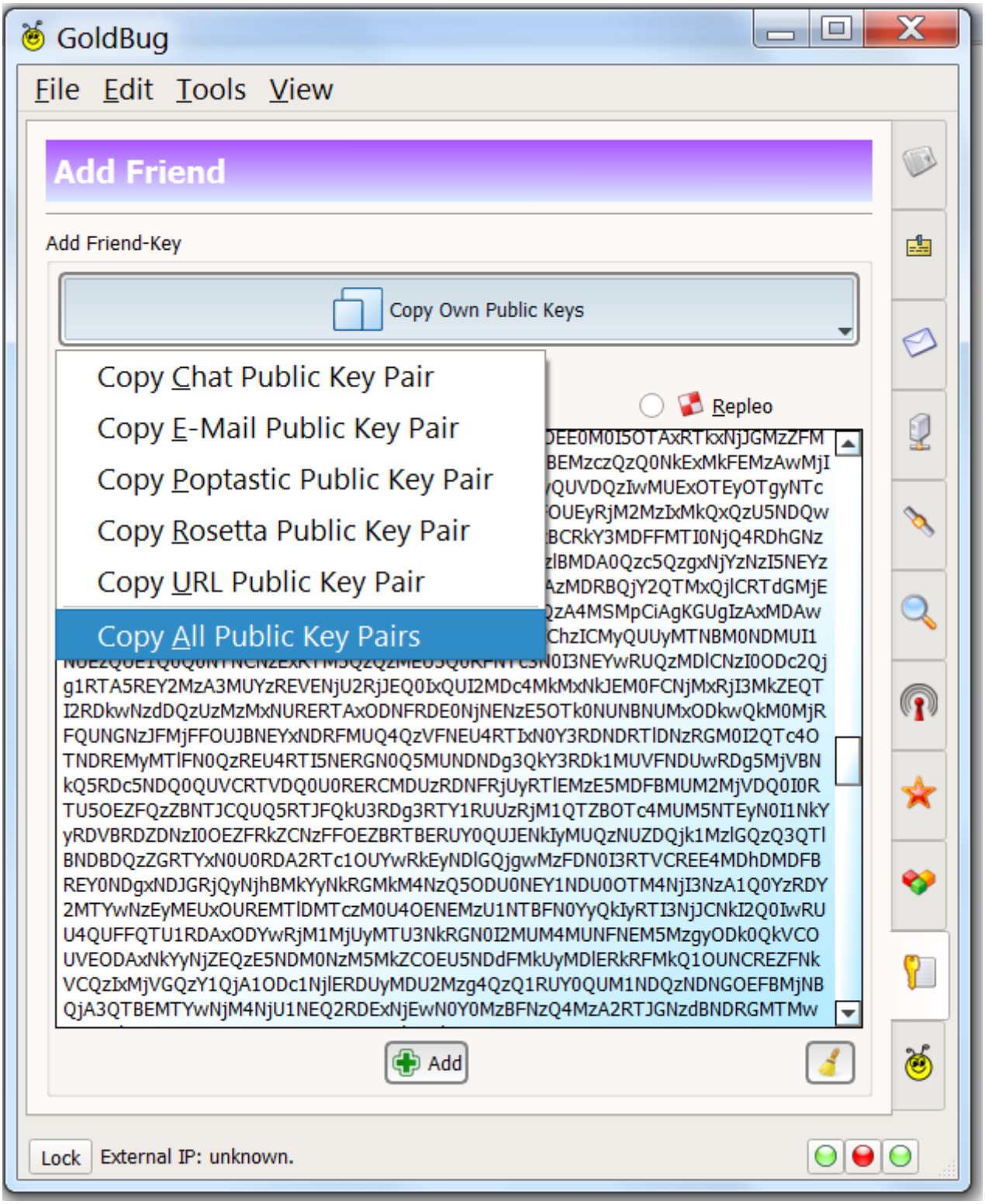
The key exchange can be described as follows:

6.1 Add a friend by swapping and inserting the keys

GoldBug uses a public/private key infrastructure, as it is well-known in the case of encryption: The public key can be exchanged with friends and the private key remains on the user's hard disk in a re-encrypted container that is opened (mounted) by the login password – and used for the application runtime.

The user and his partner, both friends, must first exchange their public key, i.e. copy it out, and then insert the friend's key in their own tab: Add Friend/Key and confirm (see figure). The friend can send his key via e-mail or another messenger. The user then copies the key into this tab and presses the "Add" button at the bottom.

Figure: Add Friend/Key



The user also finds his own public key in the tab “Add Friend/Key”. The large button (“Copy Key”) above allows the user to copy all his (or selected feature) keys to the clipboard. The user copies the full text here and sends it to his friend. Likewise, the friend does the same and the user inserts the friend’s key in the text box.

” ‘Optionally only as a note:’ ” It may be necessary to confirm a new friend as friend with the right mouse button in the context menu (make-friend function). This is used when a friend sends his key online to the user in a direct IP connection. This feature is provided in the Spot-on interface, but in the GoldBug interface, this is not available so ideally both will always simply copy, email and paste the key for each other. But if a friend uses e.g. the spot-on client with the local user interface and builds a direct IP connection to a user of the GoldBug client, then it is also possible to transfer the key via a direct IP connection instead of copy/paste. Then, the friend appears with his nick name in the tab chat or e-mail (with a different icon) and can be confirmed as a friend with the right mouse button from the context menu.

This is a further development of the Repleo, i.e. the function of encrypting your own key with the friend’s public key (upon receipt) before the return transmission starts. The key exchange is thus automated: a synchronization process follows. The user must agree that the key will be displayed after synchronization via the neighbor connection in their own client respective their own friend list.

” ‘Other option - only as a note:’ ” In addition to sending the key online via the direct IP-connection to a friend, the echo public key sharing (EPKS) tool described below can also be used. This is used if the friend is not connected to a direct connection (e.g. both partners use a shared chat server or a node is in the middle). Both partners then enter a common password secret in EPKS and send their public keys to the echo network via this password. See the more detailed information in the section of this tool, which may be a good alternative to the often uncomfortable and insecure usual key servers. This innovation by Repleo and the synchronization of the keys via the so-called Echo Public Key Share function (EPKS) or via a existing IP-connection has later also been taken up (copied) by other projects under the name AutoCrypt or KeySync. These functions are therefore based on the Repleo, EPKS and the key exchange via IP-connection of echo nodes.

6.1.1 Special feature: Repleo

If the user has already received a key from his friend (e.g. the chat key) and inserted it into his client, but now does not want to disclose his own public (chat) key to the public, does not want to transfer and store it in an e-mail program (although the public key may actually be public), then the user can also encrypt his own public key with the received key of his friend. This is called REPLEO. Hence, the key is transmitted encrypted, as soon as a user has already received a public key of the other party.

This process then has to be carried out for each function or key, i.e. the user can in each case send back the chat repleo, the email repleo and the URL repleo. The friend can also insert a

Repleo in the box of the “Add Friend/Key” tab. Above the Insert box, just define the Radio Select button, whether it’s a Key, a Repleo, or an email address you’d like to add. Meanwhile, the K and R radio buttons in GoldBug have disappeared because the client automatically detects if it’s a Key or a Repleo.

The text of a key always starts with a letter “K” or “k” and a Repleo starts with an “R” or “r”. You can still recognize it.

6.2 Start a first secure chat

The user finds his chat friend in the tab “Chat” after a successful key exchange. For the chat to work, both parties should ideally use the same and most up-to-date version of the program, generate and exchange their keys, and connect to a network node or chat server (neighbor) on the web. When the first two LEDs in the status line below light green and the friend’s name appears in the chat tab, it looks good.

If the friend’s online status turns blue (absent), red (busy), or green (ready to talk), the chat can start. Either the user marks the friend in the table and chats out of the tab, or he double-clicks on the friend and a pop-up chat window opens for that friend.

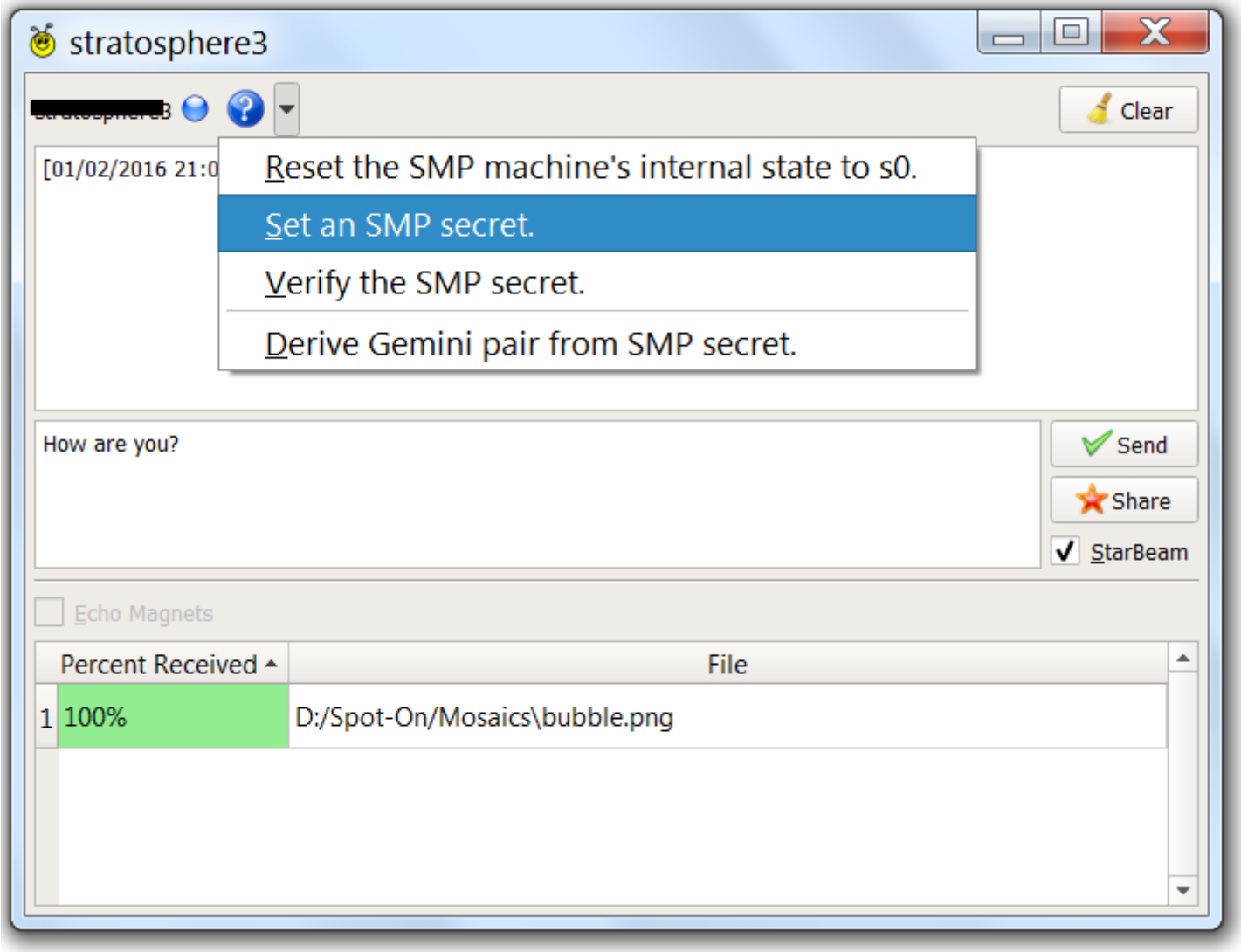
The advantage of chatting in the chat tab is that you can mark multiple friends so that the message reaches all friends. If the user uses the pop-up chat (see picture), then the user no longer needs to look at the marker to select their friend from the friends list in the chat tab.

Figure: 1:1 chat in the pop-up window

And: In pop-up chat, the user has the options button “Share StarBeam”, with which he can select a file from the hard drive, so that it is then encrypted and securely transferred to the friend (see also the section below about StarBeam-FileSharing). This feature, which allows a chat friend to simply send a file by mouse-click and is fully encrypted for end-to-end transport, is not included in many applications. Encrypted transmission e.g. of a ZIP with vacation pictures to own siblings becomes thus quite simple and is possible without the use of a hosting platform in the Web.

In the status line at the top of the pop-up window, the user can see the nickname and online status and, for example, launch the Socialist Millionaire Protocol (SMP) to authenticate a friend and test whether he knows a common secret and enters it correctly, as it will be described below. Both users will be authentic if they enter the same password with SMP.

Figure: Chat in the pop-up window

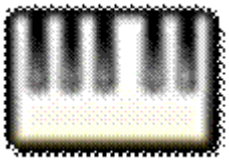


6.3 Additional security feature: MELODICA: Calling with a Gemini

MELODICA stands for “Multi-Encrypted LOng DIstance Calling” – that means: “Multiple-Encrypted Calls over a Long Distance”

The MELODICA symbol is therefore also a keyboard of a musical instrument.

Picture: MELODICA symbol



The MELODICA button performs the calling function. The Cryptographic Calling has been developed by the Spot-On kernel project and secures the connection via an immediately renewed end-to-end encryption by transmitting the password via the symmetrical connection of the echo protocol. Cryptographic Calling with the MELODICA button means calling a friend like with a phone - only that it creates a secure end-to-end encryption.

The end-to-end passphrase - also known as Gemini - is mapped through an AES string and should be kept secret between both parties. Therefore, it is important to secure the electronic transmission always very secure with further encryption levels (as here in the echo protocol with the asymmetric chat key and the TLS/SSL connection), if the transmission can potentially be eavesdropped.

6.3.1 Asymmetric Calling

GoldBug has solved this end-to-end password transfer question by encrypting the Gemini (to be formed string for symmetric encryption) asymmetrically (using the key for chat) and then encrypting again (asymmetric) the SSL/TLS channel, over which it is transmitted.

Gemini is the Greek term for Twin, meaning it refers to both participants who should then know the passphrase.

This function thus generates a “call”, a call in which the password is transmitted, which then later forms the end-to-end encryption. Strictly speaking, the Gemini consists of two keys or components, because the Gemini is authenticated by another process: This further component is also called MAC-Hash, as explained above.

The “Cryptographic Calling” as an executable protocol with the MELODICA button thus extends the previous paradigm of Forward Secrecy as follows:

6.3.2 Instant Perfect Forward Secrecy (IPFS)

The user can thus renew the (symmetric) encryption or the Gemini at any time. This means that the paradigm of “perfect forward secrecy” has been extended by two components: on the one hand, one can manually or automatically define the end-to-end passphrase (the Gemini) and, on the other hand, renew it immediately, i.e. “instant” at any time. Therefore, we speak of “Instant Perfect Forward Secrecy” (IPFS).

By comparison, many other applications offer only one key per online session and you cannot manually and individually edit the symmetric end-to-end encryption phrase.

The Instant Perfect Forward Secrecy (IPFS) discussed here uses asymmetric encryption (of the chat key), whereby the temporary key is a symmetric key (just the Gemini, an AES string).

6.3.3 Symmetric Calling

Another option is GoldBug's innovative ability to send a new Gemini through the channel of an existing Gemini. Here, the end-to-end key (that is, the symmetrically-encrypting Gemini) is sent through another end-to-end Gemini connection (i.e., the new symmetric key is communicated through a channel of an existing symmetric key). The symmetric encryption phrase (the Gemini or the AES password) is therefore not encrypted with asymmetric encryption (the chat key) (e.g. with RSA, Elgamal, McEliece or NTRU) and then sent over a secure channel (SSL/TLS) from point-to-point, but it is itself encrypted with the existing (symmetric) Gemini and then sent by the described method (again via SSL/TLS).

The double-rachet method, in which the key of the following message is in the encrypted content of the previous packet, may have been ajar or derived from symmetric calling.

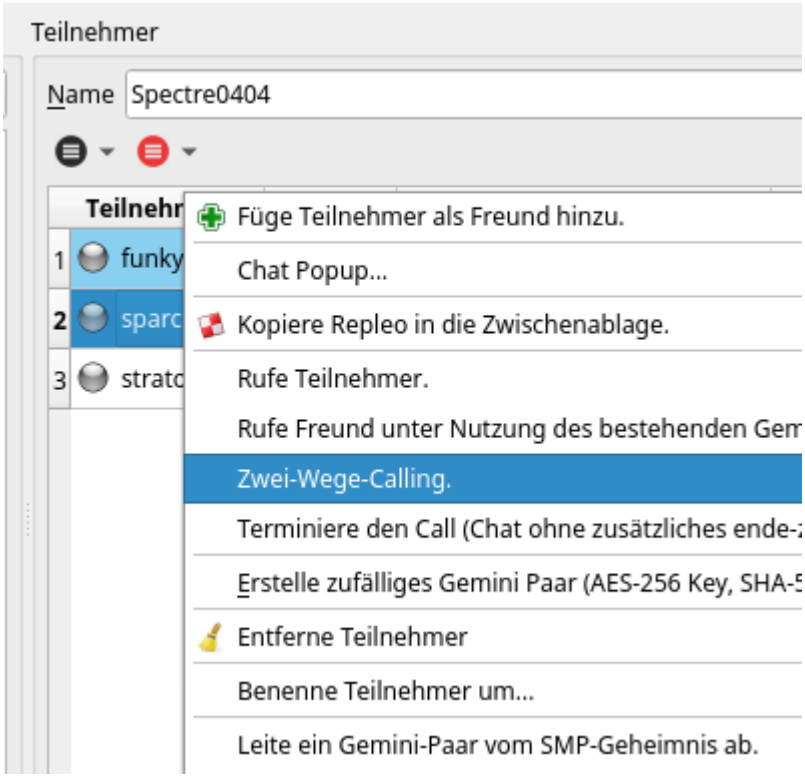
Thus, asymmetrical calls and symmetric calls can be fundamentally differentiated. Symmetric calls use an existing Gemini. Asymmetric calls send the Gemini over the asymmetrically encrypted connection (namely the permanent chat key) to the friend. Even when calling over an existing Gemini, the sent Gemini can be instantaneously renewed at any time.

In sum: Secure end-to-end multi-encryption arises when a messenger encodes a manually-defined symmetric key with an existing symmetric key and then encrypts it with an asymmetric key. And then this package is sent through a secure connection.

6.3.4 Two Way Calling

Finally, in the context menu (right mouse click on a friend in the friend list), a third method for a so-called "Cryptographic Call" is added: Two-way calling. Here, the user sends an AES-256 as a passphrase for the future end-to-end encryption to the friend, and the friend also sends an AES-256 to the first user in response. Now the first half of the AES of the first user and the second half of the AES of the second user are taken, respectively, and assembled into a common AES-256. This names the method of 2-way safety. This ensures that no third party - if someone succeeds in compromising his friend's machine - sends a Gemini (or an old Gemini) in his name from a third, foreign machine (which is not really possible, since it would mean the unnoticed acquisition of a machine or breaking the existing TLS and RSA (or NTRU or Elgamal) encryption). The two participant's ping-pong game in two-way calling ensures that both participants are currently doing their part to agree on a secure end-to-end password - Fifty-Fifty.

Figure: 2-Way Calling in the context menu from the friends list



The possibility of the password

- first, to be edited manually,
- secondly, to be able to renew every second - or within each call
- third, to send the password through an existing end-to-end encryption,
- and finally, being able to generate the end-to-end password in a two-way process makes it very difficult for attackers to break the end-to-end encryption of the GoldBug Calling feature.

“Perfect Forward Secrecy” (PFS) has become not only “Instant Perfect Forward Secrey” (IPFS), but (in this feature) even a “2-Way Instant Perfect Forward Secrecy”: 2WIPFS. This feature has significantly advanced FS and PFS and the important element of end-to-end encryption with this process implementation. The encryption itself is not new, but only the process is sophisticated implemented to provide more security.

End-to-end encryption in the GoldBug is as simple as making a phone call simply by pressing a button: just pick up or hang up the phone. At any time, the communication remains asymmetrically encrypted and the symmetric end-to-end encryption can be easily added - and also renewed by asymmetric or symmetric encryption (within an SSL channel). This is a new architectural implementation standard established by this method of Crypto-Calling.

6.4 Additional security feature: Socialist Millionaire Protocol

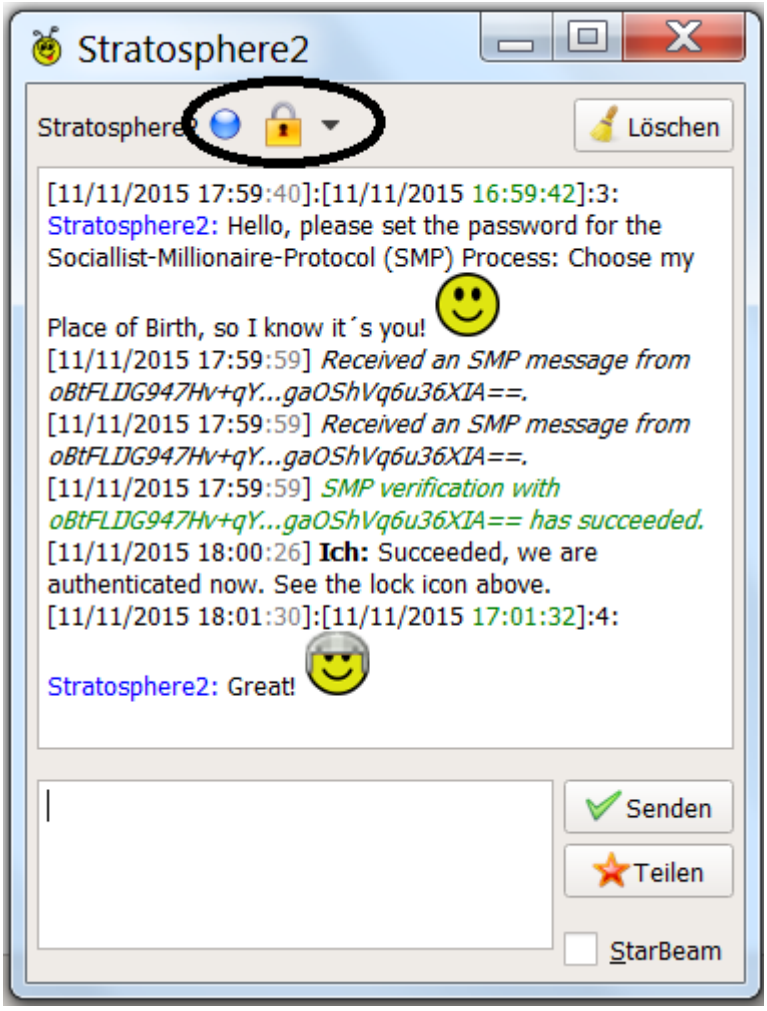
While GoldBug encrypts the messages three times -

- on the one hand the message is indeed sent in a secure TLS/SSL channel,

- secondly, every message is encrypted asymmetrically (e.g. with RSA, NTRU, McEliece or Elgamal, i.e. with the chat key),
- and third, yes, it is possible to use the “Call” function to send a Gemini to set a symmetric end-to-end encryption passphrase (as seen with different methods to perform the “calling”). E.g. within an existing symmetric encryption or via the two-way call function, where each defines half of the end-to-end password),
- Fourth, there is an additional security enhancement mechanism: it is the “SMP” protocol: Socialist Millionaire Protocol (a method also described here for off-the-record messaging (OTR): https://otr.cypherpunks.ca/Protocol-v3-4.0.0.html).

The idea behind this is to ask a friend a question like: “What is the name of the city we visited together last year?”, Or to ask a question like: “What is the name of the restaurant, in which we met for the first time?” etc. (see figure).

Figure: SMP protocol



Both participants usually sign the messages with a RSA (or other) algorithm to verify that the key used is from the original sender. But for the (possibly unlikely) case that a machine would be hacked, or if the encryption algorithm were broken, the Socialist Millionaire Protocol (SMP) process can simply identify a friend by entering the same password on both sides. It is important to ensure that the password is not to be sent through the chat, instead you should

describe a situation that leads to the same password. If the SMP process is tested for the first time, you can also enter the password “test” on both sides (in lower case).

It is practically applied as follows: The user opens a personal pop-up chat window to use SMP and clicks the question mark icon next to the user name of the chat friend. Then a password is defined with the menu. Then the chat friend is asked to enter the same password. Third, the first user then finally clicks on the Verify button.

If both participants have set the same password - or the same hash value has been generated by the same password - then the question mark icon changes to a “lock” symbol. The chat friend has now been authenticated and the chat remains safe.

SMP is thus another ideal way to authenticate the chat friend with a shared secret in the live process, so it is not additional encryption!

An example illustrates the calculation process of this protocol as follows. Let’s assume in an example that Alice begins the exchange:

” ‘Alice’ ” Records the random exponents a2 and a3 Send Bob g2a = g1a2 and g3a = g1a3

”’Bob:’” Takes the random exponents b2 and b3 Calculates g2b = g1b2 and g3b = g1b3 Calculates g2 = g2ab2 and g3 = g3ab3 Records the random exponent r Calculates Pb = g3r and Qb = g1r g2y Sends Alice g2b, g3b, Pb and Qb

” ‘Alice’ ” Calculates g2 = g2ba2 and g3 = g3ba3 Records the random exponent s Calculates Pa = g3s and Qa = g1s g2x Calculates Ra = (Qa / Qb) a3 Send Bob Pa, Qa and Ra

”’Bob:’” Calculates Rb = (Qa / Qb) b3 Calculates Rab = Rab3 Checks if Rab == (Pa / Pb) Send Alice Rb

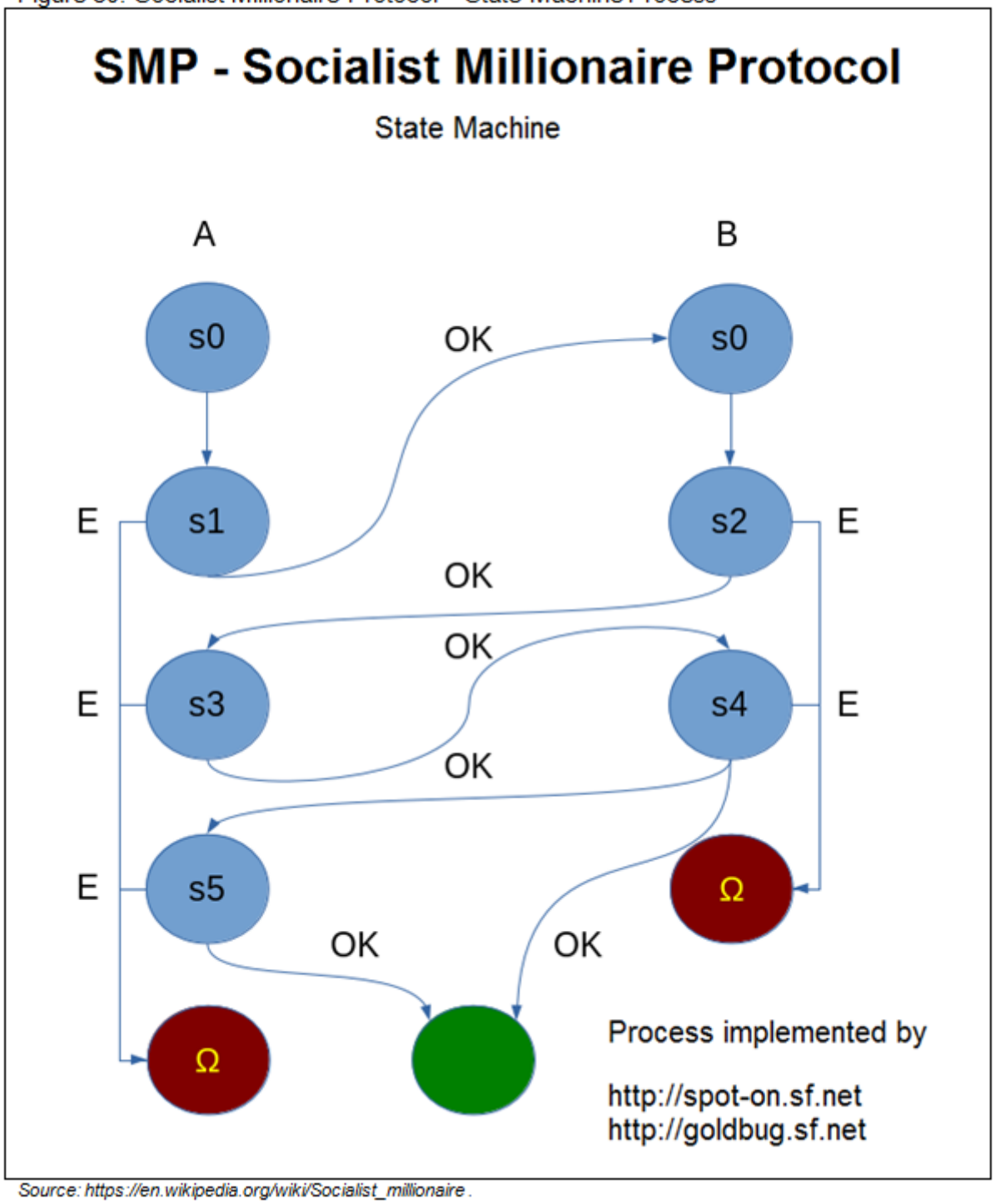
” ‘Alice’ ” Calculates Rab = Rba3 Checks if Rab == (Pa / Pb)

If everything is done correctly, then Rab should get the value of (Pa / Pb) times (g2a3b3) (x - y), which means that the test at the end of the log will succeed only if x == y. Further, no further information is revealed than that g2a3b3 is a random number that is not known to any page if x is not equal to y! (See also the formulas in the documentation of the source code).

GoldBug describes a so-called “zero-knowledge proofs” during SMP’s various data exchange processes. GoldBug also uses the SHA-512 of the entered secret passphrase as the x and y components.

Figure: Socialist Millionaire Protocol (SMP) in chat window to authenticate the chat partner

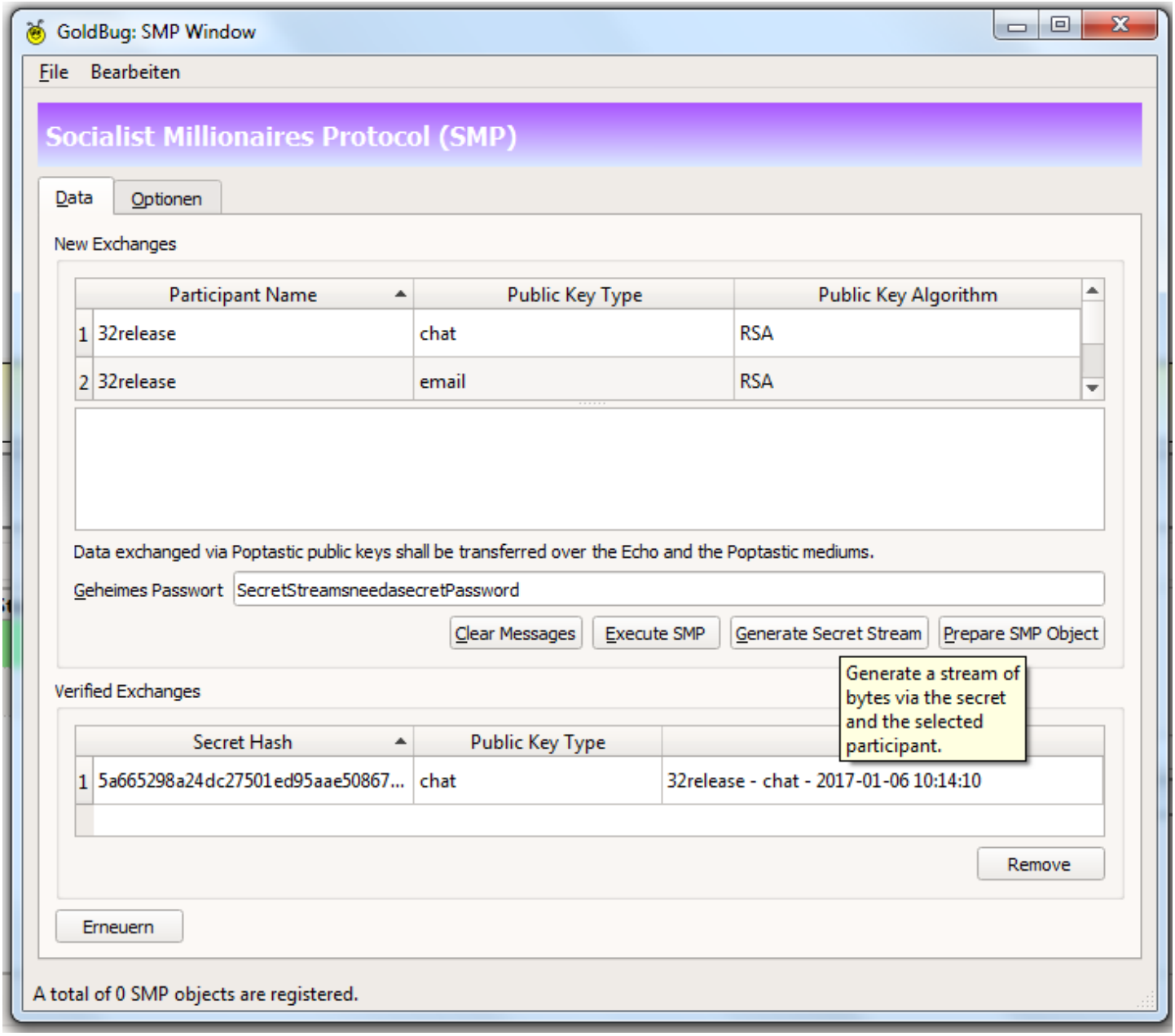
Figure 30: Socialist Millionaire Protocol – State Machine Process



Source: https://en.wikipedia.org/wiki/Socialist_millionaire .

SMP therefore requires sharing a common secret with its communication partner. If the SMP password is present, it can also be used as a basis for other functions. The secret streams function relevant for forward secrecy (see also explained at the e-mail function below) can be derived from this.

Figure: Implementation of secret streams in the extended SMP protocol



The Secret Streams are further explained in the section under E-Mail, we are here in the chat section first in the function of crypto calling, which can also build on the successfully verified SMP password. This is described by SMP-Crypto-Calling.

6.4.1 SMP-Calling

Above, we explained the call function of how to generate and transfer a Gemini. A User can not only define the Gemini manually or through the AES function, but it can also be derived from the password stored in the SMP process as outlined above. Thus, the password input from the SMP process is used (not the SMP process itself). It is another way of “crypto calling” and thus securely transmits to its counterpart an end-to-end password that does not originate from an AES generator this time! - if someone doubts the randomness of a machine number generator. Once the basic functions of encryption in GoldBug are clear, you can see, for example, the interconnectedness of the individual processes in the architecture.

6.5 Additional security feature: Forward Secrecy (asymmetric)

Since version 2.7, GoldBug Messenger has also been supporting Perfect Forward Secrecy for its role as an e-mail client, making it the first e-mail client to offer Forward Secrecy for e-mail, with both symmetrically and asymmetrically Forward Secrecy (see also further down).

While crypto calling with a Gemini for the chat function has the “instant perfect forward secrecy” and refers to a symmetric key (just the Gemini or the AES string), the perfect forward secrecy is in the email with temporary, a-symmetric keys defined.

This variant of the use of temporary asymmetric keys can of course also be transferred to the chat function. And this has just been done since the release 2.7.

While chat with the permanent chat key is always (asymmetrically) encrypted, a temporary asymmetric key is now used with this new layer of end-to-end encryption. This temporary asymmetric key is called an ephemeral key. This key is created by the forward secrecy function in the chat, which is displayed via the context menu (right mouse click) or via the menu button.

A tooltip on the screen (in the systray) indicates when the chat partner in chat has created a forward secrecy with temporary (ephemeral) asymmetric keys, so that the user can confirm this in his client in a pop-up window. The user looks at the bottom of the status line for the newly appearing icon, clicks on it and can then confirm the forward-secrecy process in the appearing pop-up window. Then, the (temporary) chat key is no longer used, but the new, temporary a-symmetric keys. The permanent chat key is thus complemented by the temporary chat key.

Only few software applications understand end-to-end encryption as asymmetrical and build forward secrecy via asymmetric encryption.

6.5.1 Forward Secrecy Calling

Thus, the calling can be extended again: The symmetric Gemini is sent in the Forward Secrecy Calling (FSC) not as described above by the permanent (asymmetric) chat key or by an existing (symmetric) Gemini, but by the ephemeral, temporary and asymmetric chat key.

While sending a Gemini over an existing Gemini defines a “symmetric” “instant perfect forward secrecy”, sending a Gemini over the ephemeral keys of the initiated “forward secrecy” in the chat function may be considered an “asymmetric” one of “Instant Perfect Forward Secrecy”.

(But also sending a Gemini via the permanent chat keys is called an asymmetric “Instant Perfect Forward Secrecy”).

While “Forward Secrecy Calling” and “Call by a Gemini” already have a “Forward Secrecy” and then define the renewability of the end-to-end key at any time (Instant Perfect Forward Secrecy), the other Calling Types are not with Forward Secrecy given in advance, but Instant Perfect Forward Secrecy is generated here only by a call as a result of the call.

The continuation of Forward Secrecy is called Forward Secrecy Calling.

6.6 Overview of the different Calling types

From the methods described (see also figure), to transfer an end-to-end key to the friend, the following overview can be designed, which highlights the different methods with their respective specific characteristics.

Overview of the different Calling types with respective criteria

Figure: Overview of the different types of callings with respective criteria

Kriterium	Asymmetrisches Calling	Forward Secrecy Calling	Symmetrisches Calling	SMP-Calling	2-Wege-Calling
TLS/SSL-Verbindung	JA	JA	JA	JA	JA
Permanenter asymmetrischer Chat-Schlüssel	JA	JA	JA	JA	JA
Symmetrisches AES-Gemini als Kanal	NEIN	NEIN	JA	NEIN	NEIN
Halbes symmetrisches AES als Gemini (50 % AES + 50 % AES)	NEIN	NEIN	NEIN	NEIN	JA
Geheimes SMP-Passwort als Gemini	NEIN	NEIN	NEIN	JA	NEIN
Ephemerale asymmetrische Chat-Schlüssel	NEIN	JA	NEIN	NEIN	NEIN
Forward Secrecy als Voraussetzung	NEIN	JA	JA	NEIN	NEIN
Instant Perfect Forward Secrecy als Ergebnis	JA	JA	JA	JA	JA

The call information - that is the end-to-end encrypting passphrase - can of course also be transmitted manually, e.g. verbally or by telephone. If one adds the above-mentioned existing call types, it concludes then in total in six different ways to be able to implement a call. For the first time, the spot-on architecture has spoken in the crypto discipline of “Crypto Calling” for the transmission of end-to-end passwords, and later concepts have borrowed this term.

Please note the following explanations:

- Each of the presented methods results in Instant Perfect Forward Secrecy (IPFS).
- Only symmetric and asymmetrical calling requires no action on the part of the other party.
- Forward Secrecy Calling and Symmetric Calling require an existing status of Forward Secrecy.
- ‘Symmetric Calling’ and ‘Forward Secrecy Calling’ have triple encryption layers (TLS/SSL, Permanent Chat Key, as well as temporary symmetric or asymmetric key through which the new Gemini will then be sent).
- ‘SMP Calling’ and ‘2-Way-Calling’ break AES generation by replacing parts of the AES phrase or creating a new password string.

The message formats with the encryption levels then look simplified - since signatures, HMACs and hashes are not included - as follows:

- Asymmetric Calling: (TLS/SSL (Permanent Chat Key e.g. RSA (message is an AES string)))
- Symmetric Calling: (TLS/SSL (AES (Permanent Chat Key e.g. RSA (message is an AES string)))

- Forward Secrecy Calling: (TLS/SSL (Permanent Chat Key e.g. RSA (ephemeral keys RSA (message is an AES string))))
- SMP calling: (TLS/SSL (permanent chat key e.g. RSA (message is a string formed from the SMP)))
- 2-way-calling: (TLS/SSL (Permanent chat key e.g. RSA (message is an AES string that is 50% modified with friend's AES)))

From this variety of options in securing end-to-end encryption or even defining and manually entering the end-to-end encryption passphrase, the slogan, claim or headline for GoldBug has emerged: "Your Instant Definition in Decentralized Crypto". The encryption is thus not only a user-specific, which can be renewed (instant) at any time, but also a decentralized at the user's place - defined and designed by himself.

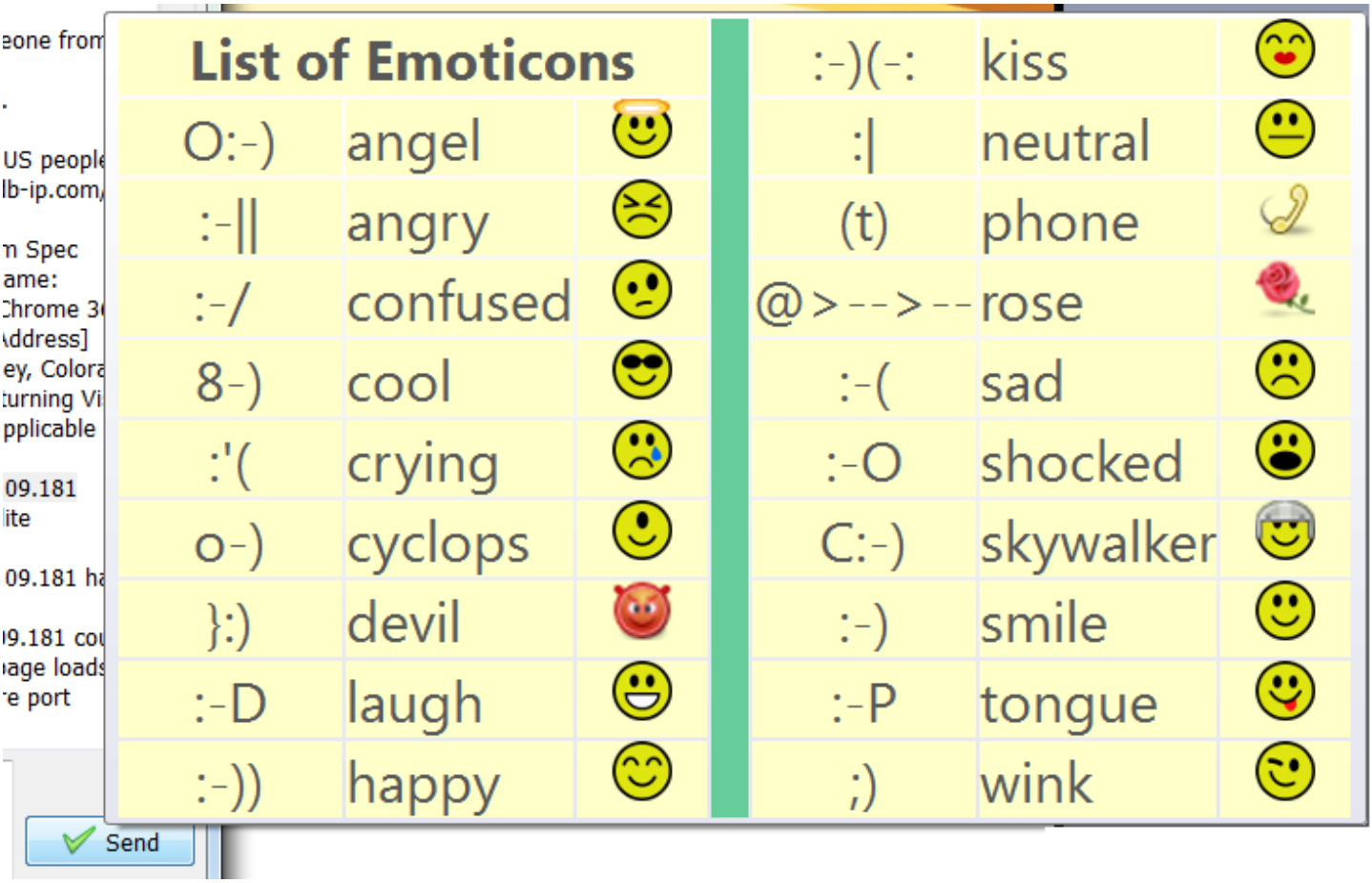
Figure: GoldBug Claim: Your Instant Definition in Decentralized Crypto

A simple litmus test compared to other software applications is the simple question of whether the user can enter the end-to-end encrypting password himself. With GoldBug the user can do it (as well as with the mobile version of GoldBug: Smoke Chat for Android).

6.7 Emoticon

GoldBug offers a variety of different emoticons - also called smileys - for chatting (see picture).

Figure: Emoticon list in GoldBug Messenger



To use them, the user clicks twice on a friend, so a pop-up chat window opens for private chat. If the user now moves the mouse over the Send button, the smileys are displayed in a tooltip that appears. By entering the ASCI code, the emoticons are then displayed in the chat.

In the chat tab, the options of the right-hand page splitter also allows the user to turn off the graphical display of smileys in general.

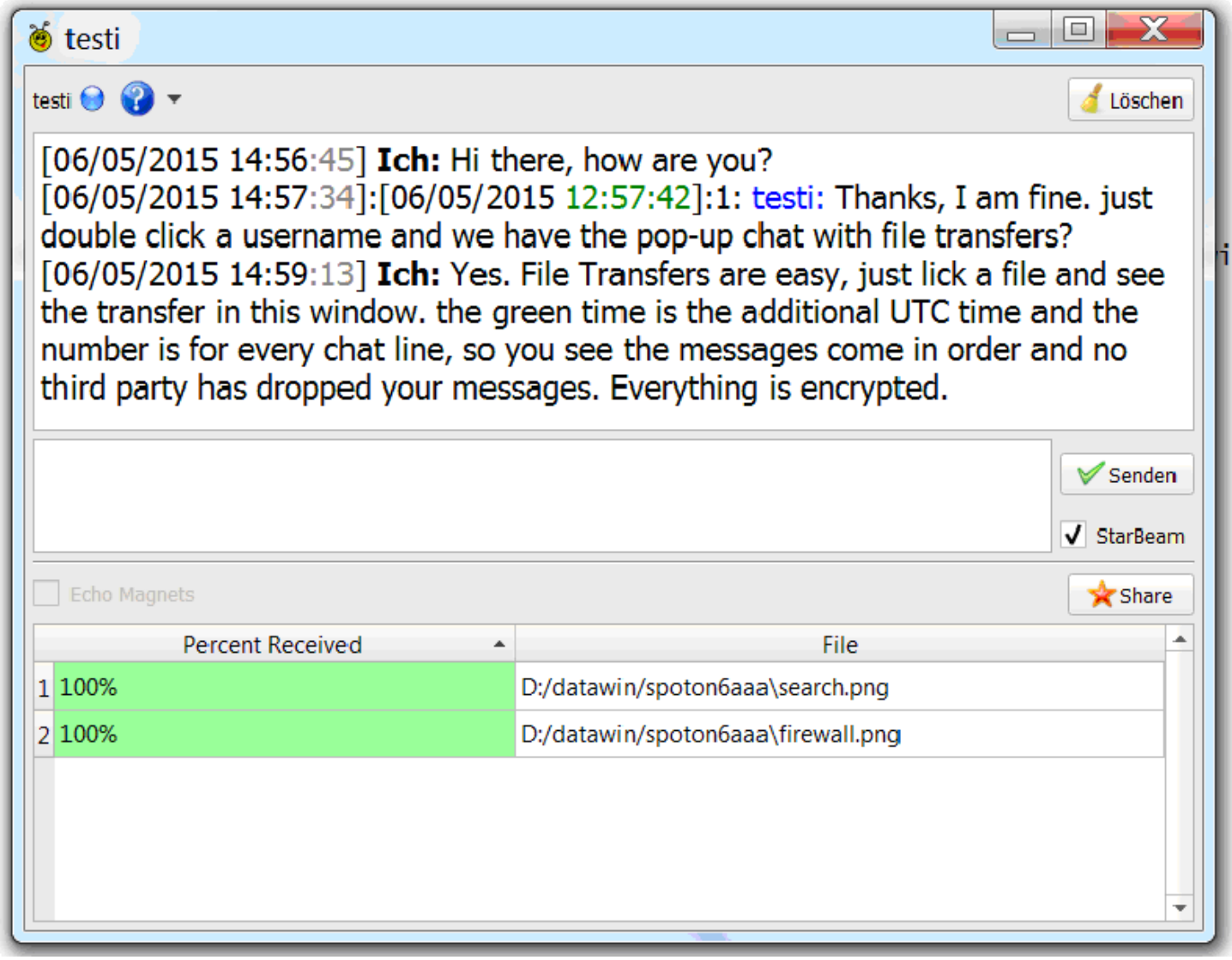
6.8 File transfer in the chat pop-up window

The Qt menu allows to remove individual menu parts from the regular user interface and to create a pop-up window.

Figure: Tear-off / hook-up of controls

Likewise, the file-sharing function in particular is integrated in a pop-up menu: In the 1:1 chat window. So if a user wants to send a file to a specific friend, they can simply click the button in the pop-up chat window with that friend.

Figure: File transfer in the chat window

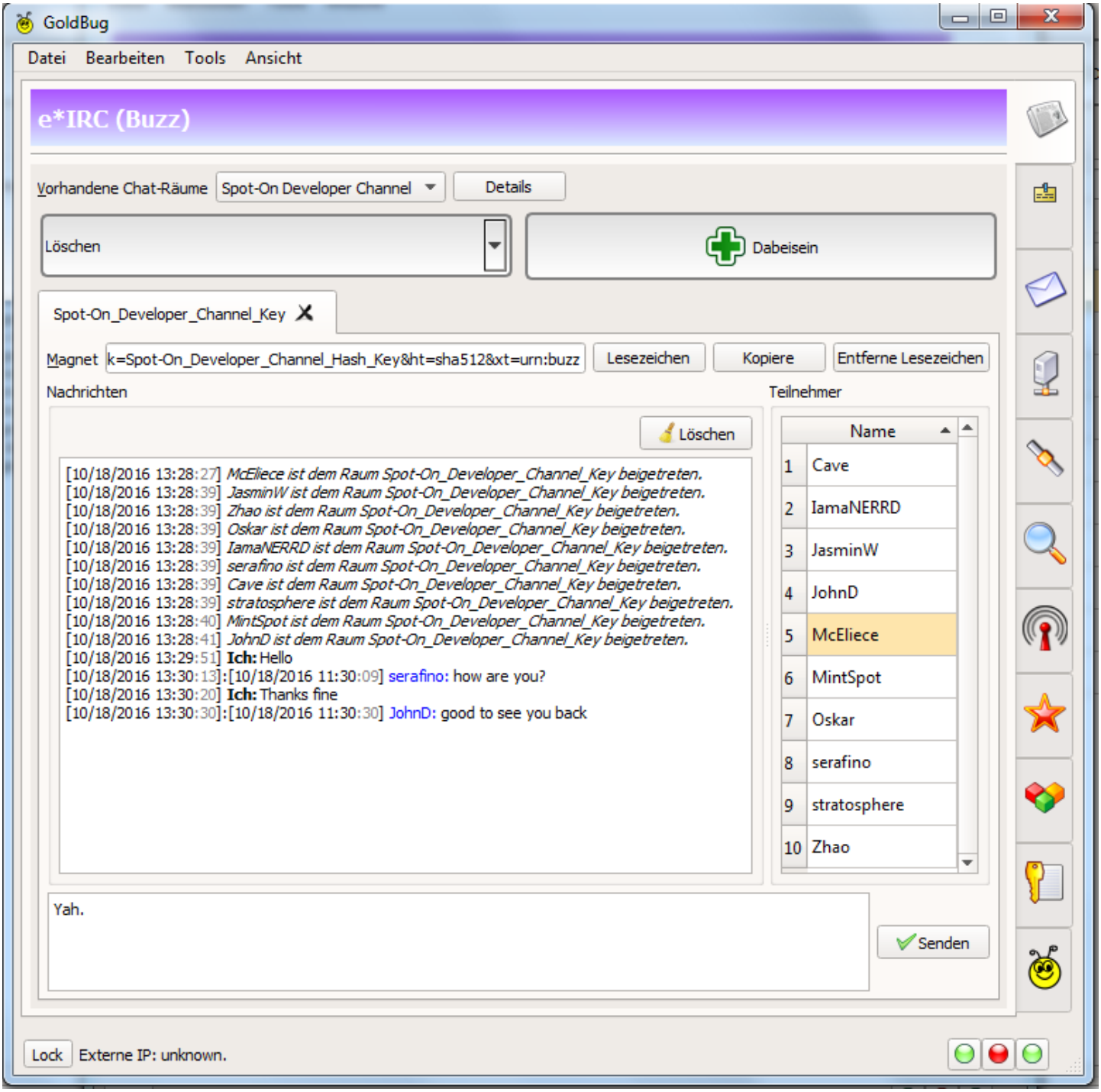


The file as well as the text is transmitted securely and encrypted to the friend. The file transfer feature is called StarBeam and also has its own tab, but is already built into the 1:1-chat-window for easy and direct usability. Just another little hook-up menu.

7 Group chat in IRC style

In addition to email and chat and transfer files to the communication partners, GoldBug Messenger also has a group chat feature. This works similar to an IRC chat. The transmission of the messages to all group participants is again fully encrypted via the echo protocol. The encryption is symmetrical, similar to a password string. Finally, in the p2p network or via the chat server, all subscribers can read a group chat that knows a particular symmetric end-to-end key that defines the chat room. The group chat is also based on the echo protocol.

Figure: The e'IRC group chat



It is therefore spoken of echo'ed IRC (or in short e'IRC). That opens up new options for the IRC chat, since the transport routes of the e'IRC chat are also encrypted.

As normal POP3 or IMAP e-mails today also have at least one transport encryption, e.g. with TLS 1.3, IRC can also be understood as an encrypted group chat. Also, the traditional IRC chat can therefore learn from such security models and improve: The e'IRC chat can represent a model of a new group chat generation.

The encryption details of the group chat are again defined via a magnet URI link (see below) (defined in the link with extension &URN=buzz). Buzz is the technical name in the source code for the e'IRC group chat.

To start the GoldBug program, open the as preset given community chat room, which can serve as an example. Here, the user can ask the other present users questions about the program or

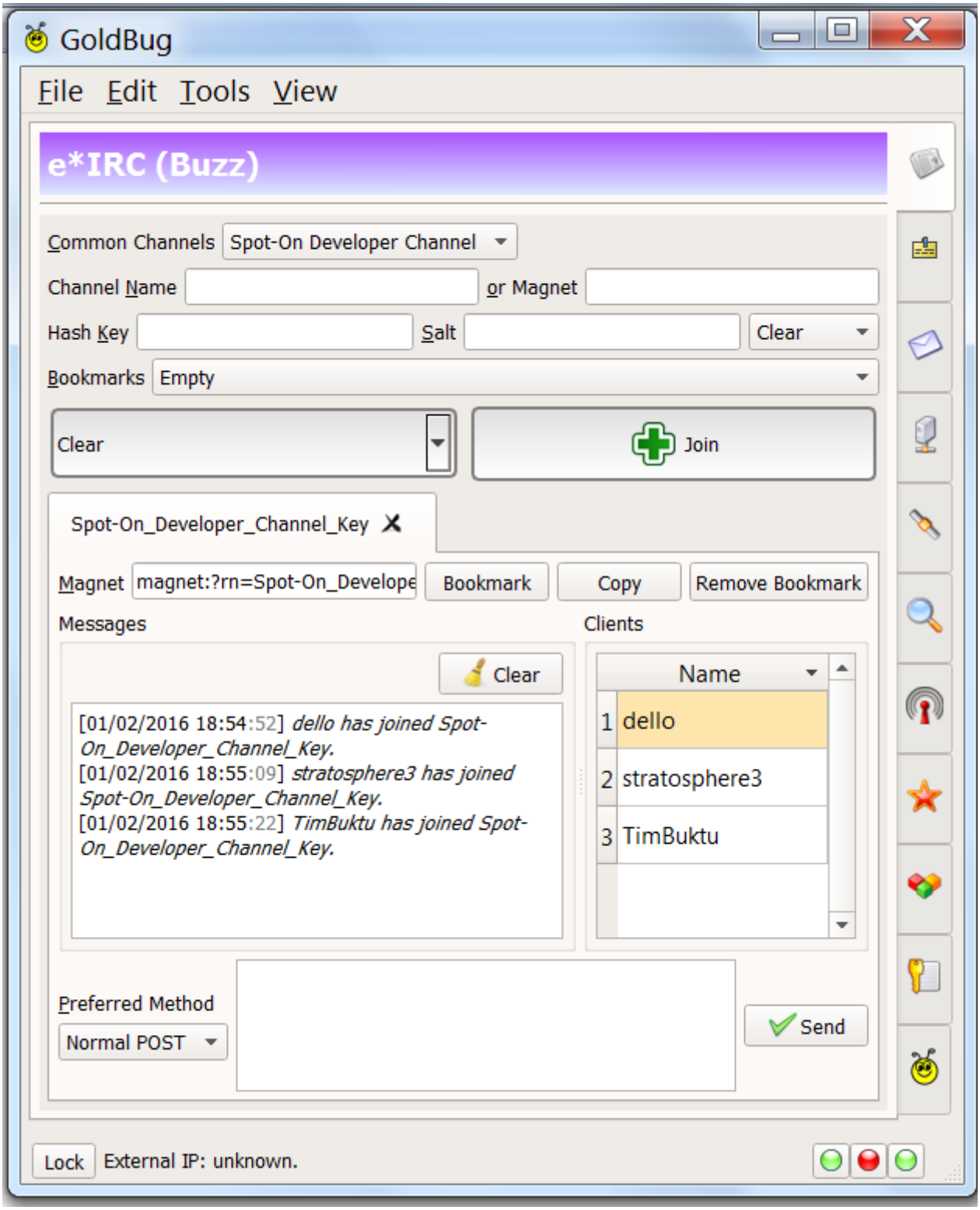
just use this channel with a friend.

To join an own channel, the user simply enters a room or channel name or uses the magnet link method above. The magnet link has embedded additional values for encryption in addition to the room name, such as key, hash or cipher for the encryption type.

If the user enters only the room name and does not use a magnet URI, the additional encryption details are set to the value 0000 and the encryption of the room is based on the room name. If the user has entered all the values or the room name or the magnet link, then the “Join” button is just to be pressed.

If the user has inserted a magnet as link, then in the pull-down menu the command “de-magnetize” should be used first. The magnet is then broken down into its individual components and the room is created and entered based on the encryption values embedded in the magnet link.

Figure: IRC-style group chat within the e’IRC buzz channel



If the room is open, the user can also save the room as a bookmark or copy the corresponding Magnet URI at any time from his chat room as a bookmark and send it to his friends to invite them to the room.

In order to send a message, the user then enters a text in the chat room and presses the send button.

The e'IRC chat room can be public or private, depending on how much the user announces the magnet or the individual encryption values. As a public e'IRC chat room, the user can post or link the magnet URI on the own website and everyone knows how to get into that chat room: with "de-magnetize".

Ultimately, it works for these news channels like an IRC chat, only with the difference that the Internet provider and other rooting server cannot look into the communication, since it is encrypted - as a connection in online banking too.

So it does not make any difference whether a user is talking to friends or his online bank advisor.

If the user wants to use the chat room as a private room, the user can secretly share the Magnet URI only with his friends and one stays on one's own. This is a convenient feature of the GoldBug program: the user can simply chat encrypted without first exchanging asymmetric keys. The user simply tells his friend verbally that he should come in GoldBug in the room "Amber Room" and both participants can very easily and securely encrypted using a common chat server.

Tip: The user can create a one-time magnet for a room. This is used to protect his public chat key when exchanging the key with the communication partner through the (self-defined) IRC channel. It requires that the magnet URI is only known to the friend.

With the Repleo, with EPKS and the key exchange via a one-time magnet (OTM) for a private e'IRC chat room, GoldBug offers several methods for a secure key transfer. Thus, public keys no longer have to be public!

8 Smoke Mobile Chat Client

While GoldBug is a desktop client that is compiled and deployed on numerous operating systems as well as platforms such as Raspberry Pi, the mobile client of the Echo protocol is called "Smoke Chat" in Java.

8.1. Smoke Android Client

Smoke offers a direct 1:1 chat to a friend as well as a group chat. This is called FireChat and is similar to the Buzz/e'IRC chat in GoldBug.

The 1:1 chat of Smoke on the mobile device does not use the phone number of the participants as an identifier, but a short string, a so-called SIP hash, is used as an identifier.

Smoke users connect to a common server - this can be a GoldBug, Spot-On, Spot-On-Lite, and SomkeStack server listener - and then swap their public key over the SIP hash connection. SmokeStack is a chat server for Android and can serve around 500 users on an Android device - ideal for a workshop group, family or school.

8.2. Fire to Buzz Chat

Since Java and C ++ programming do not know common key formats from the crypto libraries, it is usually not possible to use an open source Java client to chat encrypted to a C ++ client.

Smoke, however, has innovated and implemented a way to do so: the so-called FireChat in Smoke can also be used to reach a user in GoldBug and vice versa. This is based on a symmetrically encrypted chat (similar to a symmetric crypto call).

Other applications mostly use the Java Script Crypto libraries for the browser or connect to the central server, but these methods are generally considered to be less secure.

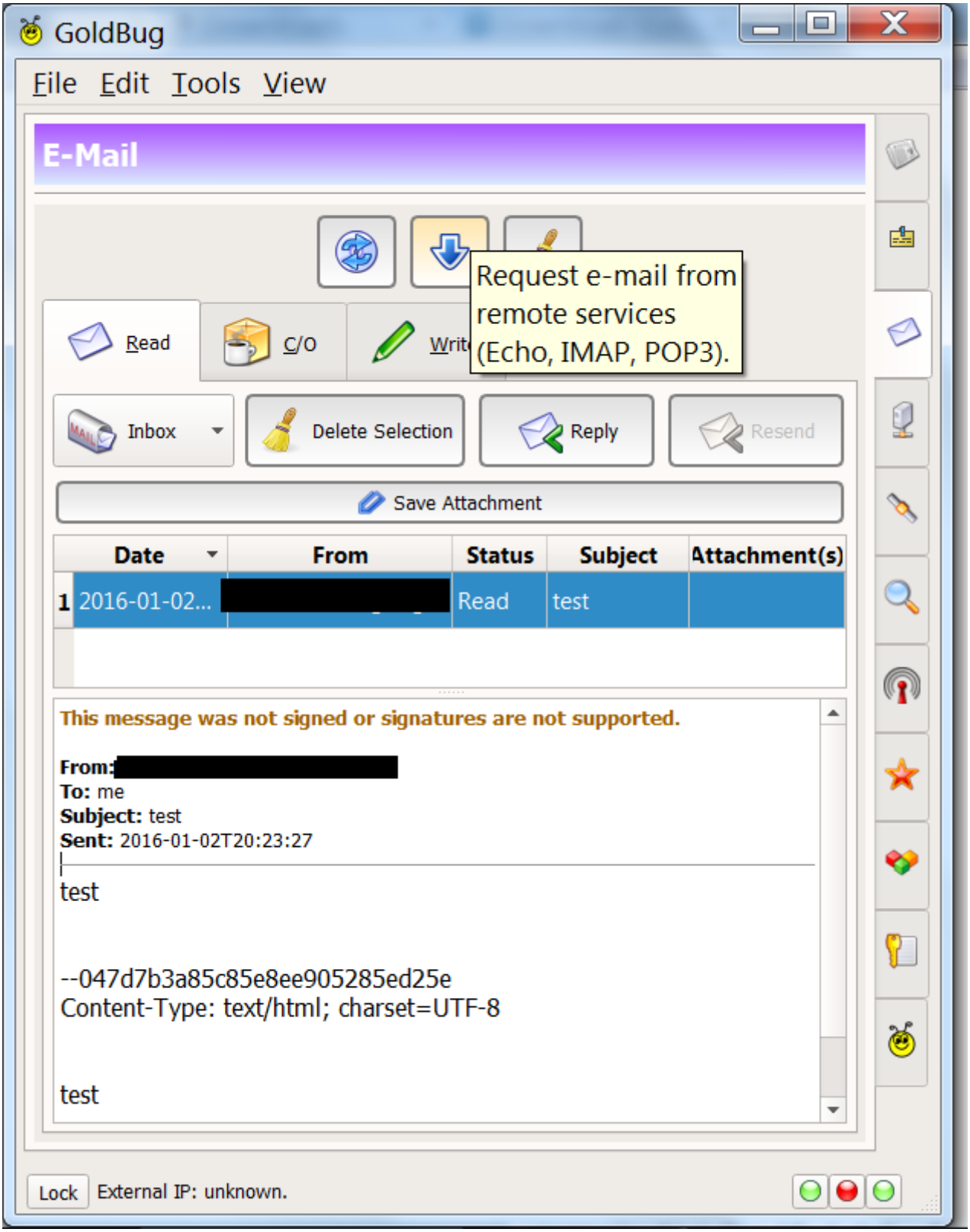
So if you want to use a mobile version of GoldBug for the chat, this can be found in Smoke Chat and under the German-language concept description MOMEDO (both at Github: https://github.com/textbrowser/smoke).

9 The e-mail function

GoldBug is a fully functional email client.

Not fully - like email applications that have existed for decades - here it still needs further programming from the community, but still fully functional in the sense of a fully usable email client. The advantage: the reading and writing of e-mails is shown very efficiently in the user interface on one page or in a tab (see figure). And: the emails to other GoldBug users are always encrypted.

Figure: E-mail - read view



Technically uses GoldBug the library CURL and supports POP3, SMTP and IMAP. Finally, GoldBug's special feature is that it also supports p2p e-mail.

Here the e-mail is stored in the distributed network of the participants and not at a central provider. With GoldBug, users can very easily provide an e-mail server and communicate with it. The infrastructure is not only easy to install but can also be created by the user.

From a future perspective, this is also the (necessary) future, that the users of the Internet organize the Internet again more on their own and use cryptography with their own encrypted

mailboxes that are not deposited at central hosters, but on their own network of participants.

After all, centralization is always followed by decentralization, even though only the users, who recognize and value this freedom, will pay attention to decentralization necessities and such remaining opportunities in the future.

Here's how to set up the three ways to load your emails:

9.1 POP3

The Post Office Protocol POP3 is a transmission protocol that allows a client to pick up e-mails from an e-mail server.

POP3 allows you to list, retrieve and delete emails on the email server. For sending e-mails, the Simple Mail Transfer Protocol SMTP is usually implemented in clients and servers as a complement to POP3.

The POP3 protocol is thus integrated in all popular e-mail programs, including GoldBug. How it is set up next to IMAP is explained below and also further below in the description of the window of POPTASTIC (see the following figure).

9.2 IMAP

The Internet Message Access Protocol (IMAP) was designed in the 1980s with the emergence of personal computers to resolve the e-mail storage on individual client computers in the mail communication.

That is, the (PC) clients instead access the information online on the servers and, if necessary, receive copies of it.

While a user of POP3 has lost all e-mails after losing his or her PC, a mail client at IMAP only copies the requests to the server for the information currently required.

For example, if a user wants to see the contents of his inbox folder, the client will get an up-to-date copy of the message list from the server. If the content of an e-mail is to be displayed, it is loaded as a copy by the server. As all data remains on the server, a "local storage of the data" is unnecessary and extended possibilities such as the search of mails are also performed only on the server side.

This also makes local backup of the data - by taking it away from the server - mostly impossible, as the configuration of IMAP by default is not geared to it. At the same time, the issue of confidentiality and security of data that is outsourced to IMAP servers comes to the fore in the case of unencrypted mail. The question arises as to whether the recipient of an e-mail has jurisdiction over the confidentiality and storage of the e-mail itself, e.g. has the right not to show it to anyone or to delete it in secret, or if he only has one copy, gets a "right of view" of his mail.

With regard to the findings from 2013 – to better encrypt emails fundamentally - IMAP is to be judged particularly critically in this light: The storage of emails is not made at IMAP as in POP3 in the mail client on the machine of the user, but the personal data remain unencrypted on the server of the provider. With IMAP, the cloud, which is widely used today, was invented in the field of e-mail in the 1980s. POP3 is more likely to enable on-premises handling of e-mail storage on the local machine.

GoldBug supports this standard as well as POP3 and makes it possible to receive and send plain text messages via IMAP. Here's how to enter the settings for an email account in GoldBug.

Detailed description of POP3 / IMAP setup options:

Via the main menu "View" of the GoldBug Messenger the own e-mail address and the POP-3 or IMAP server details are stored. These are the same details that are also entered, for example, in the Thunderbird e-mail client or Outlook, for example:

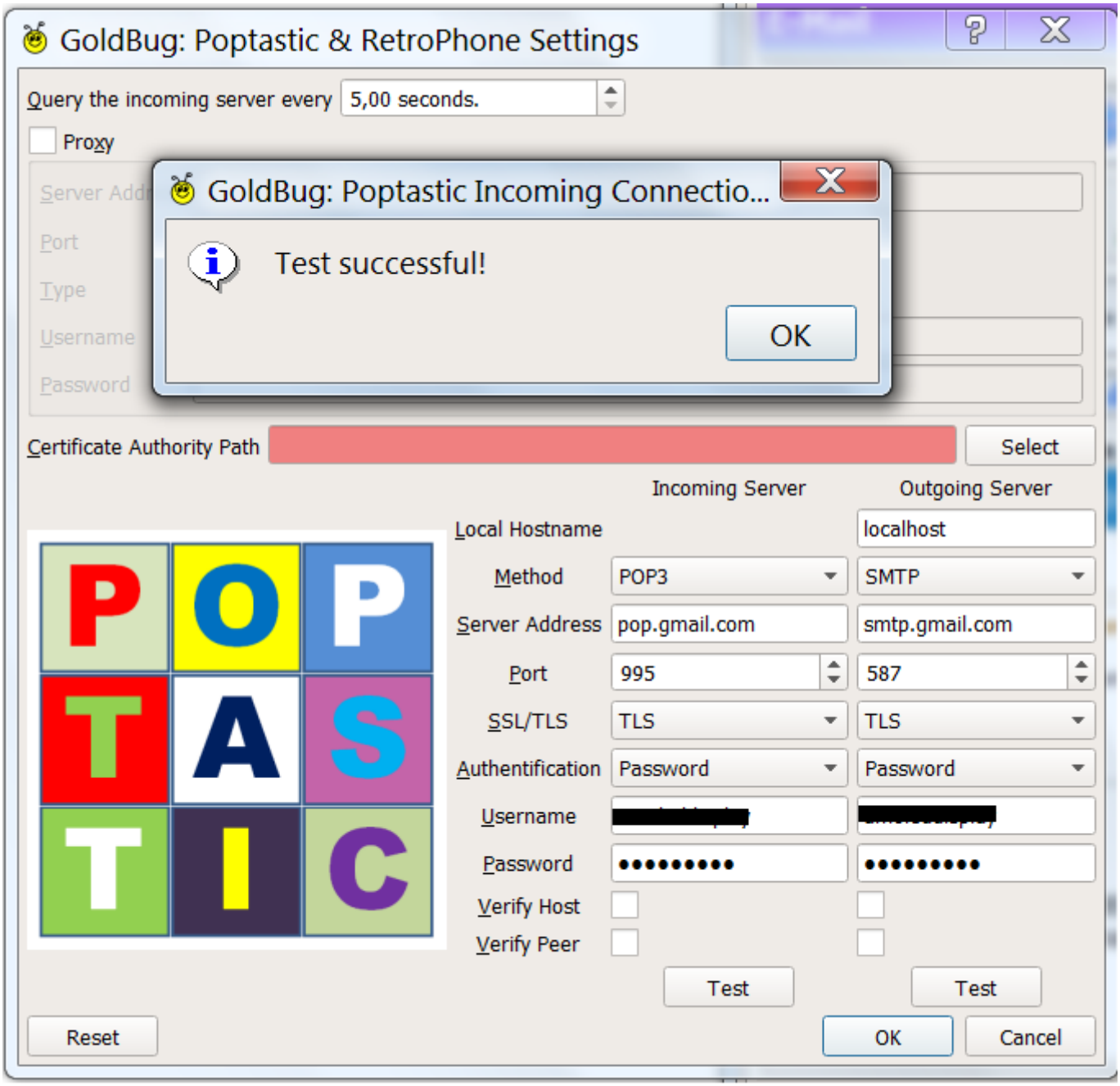
" 'Incomming Server Server:' " pop.gmail.com Port: 995 TLS Username: mygmailname@gmail.com Password: ** **

" 'Outgoing Server Server:' " smtp.gmail.com Port: 587 TLS Username: mygmailname@gmail.com Password: ** ** The user can press the test button to check the functionality of the server input. With the "OK" button then the inputs are stored.

(If the value "Disabled" is used in the selection menu instead of POP3 or IMAP, the program no longer sends e-mails: the mail function is completely switched off.)

Users who want to use the chat function described below via the POP3 / IMAP e-mail server (POPTASTIC) should therefore not deactivate the e-mail information.

Figure: POPTASTIC: chat via email server



According to the above security considerations, a user should always load his emails right from the server onto his own machine and delete them on the server. So, there seems to be much to talk about using POP3 instead of IMAP, as IMAP is more focused on keeping emails on the server.

In general, e-mails in this light do not belong to a remote server, not into the cloud, not into a browser-based web service - they are stored on the user's own machine - or they are in any case to be encrypted.

But the trend today is exactly the opposite: Central servers that store the messages, without encryption, without own infrastructure in the hands of users. This will last, until the trend reverses again and users will rediscover their own sovereignty.

9.3 P2P E-Mail: without data retention

Third, in addition to IMAP and POP3, there is the option of using p2p email in GoldBug. This means that the emails are not stored or cached in a central server, but in the client of a friend.

With regard to encryption, it has already been shown that the e-mail function uses a different encryption key than the chat function. So the user can add a friend to the chat, but refuse the email or vice versa. It makes sense, however, to copy all the keys as a whole, then the user has his friend present in all functions (so in addition the URL key, the POPTASTIC key and the Rosetta key - three functions that will be described later).

Of course, in the key transfer for the e-mail function, the security of a Repleo can be used again, if you do not want to reveal your own public e-mail key to the public.

No matter which e-mail method a user chooses, whether POP3, IMAP or P2P, outgoing emails in GoldBug are always encrypted, there is only one exception, that is if the user in the Add Key Tab does not have a Key (or a Repleo), but chooses the selection: Add E-mail-Adress. Then the e-mail program sends unencrypted text from @ -mail to @ -mail.

Note: Anyone entering a POPTASTIC key will also see the @E-mail address in the contact list for email, but it is color-coded and also has a padlock icon, which means it will be POPTASTIC email address – just used for encrypted emailed - and also chat. After all, a key is inserted for POPTASTIC and no @-e-mail address. Only e-mails sent to @-e-mail addresses that do not have a lock symbol remain unencrypted.

To clarify again:

The user can use the following ways by e-mail

- The e-mail key: This can send e-mails via POP3, IMAP and P2P.
- The POPTASTIC KEY: This can send chat via POP3 and IMAP
- The @mail address: This can send unencrypted emails and regular @mail addresses via POP3 and IMAP.

Therefore, two GoldBug users can exchange encrypted e-mails with normal @Mail, e.g. via the major e-mail providers such as Ymail, Gmail, GMX, Outlook etc. without any further technical knowledge: either unencrypted via @-mail addresses or encrypted as chat over the POPTASTIC key, which will be explained later. And thirdly, you can always use the e-mail key to send encrypted e-mails, including p2p.

This is very comfortable in that it is sufficient to exchange the keys once. So it is not every single email that the user writes each time again to encrypt individually (as previously practiced in other procedures practice). Each @ -mail provider can now be exempted from viewing the user's emails by simply pushing encrypted ciphertext over the central server to the communication partner. What is needed is the agreement with the friend that he also uses GoldBug or one of the other Echo clients as an e-mail client to exchange the keys once.

E-mail attachments can also be attached to an e-mail as a file and are automatically encrypted regardless of which encryption e-mail method is chosen. This is also possible with several attachments.

In addition to the encryption of e-mails, the meta-data is still stored in many countries, i.e. when and how often a user retrieves the messages from his mailbox. Here is the other method of p2p emails interesting:

GoldBug also makes it possible to store e-mails on the subscriber network or on its own server and decentralize the corresponding e-mail inboxes, which also exclusively and automatically handle the standard of encrypted e-mails.

The e-mail client thus also contains a peer-to-peer-based component, i.e. the e-mails are sent over the network of the encrypted connections and buffered in the nodes of friends. This network is provided by the integrated architecture of the Spot-on kernel. The advantage of P2P e-mail is that the e-mail inbox does not reside with a central host and public e-mail provider, but can be set up in the decentralized network of the user's own friends.

With GoldBug everyone is able to easily set up an e-mail inbox for their friends. Nobody can then log when and how often a user retrieves own emails. The echo protocol also helps to minimize metadata that reveals who has read which e-mail and who is storing an e-mail for whom (since the opening of the encrypted messages occurs exclusively on the user's machine and each one - according to the Echo protocol - sends each message to everyone).

How to set up a mailbox for friends is shown in the following section:

9.4 Setting up C/O & e-mail institutions

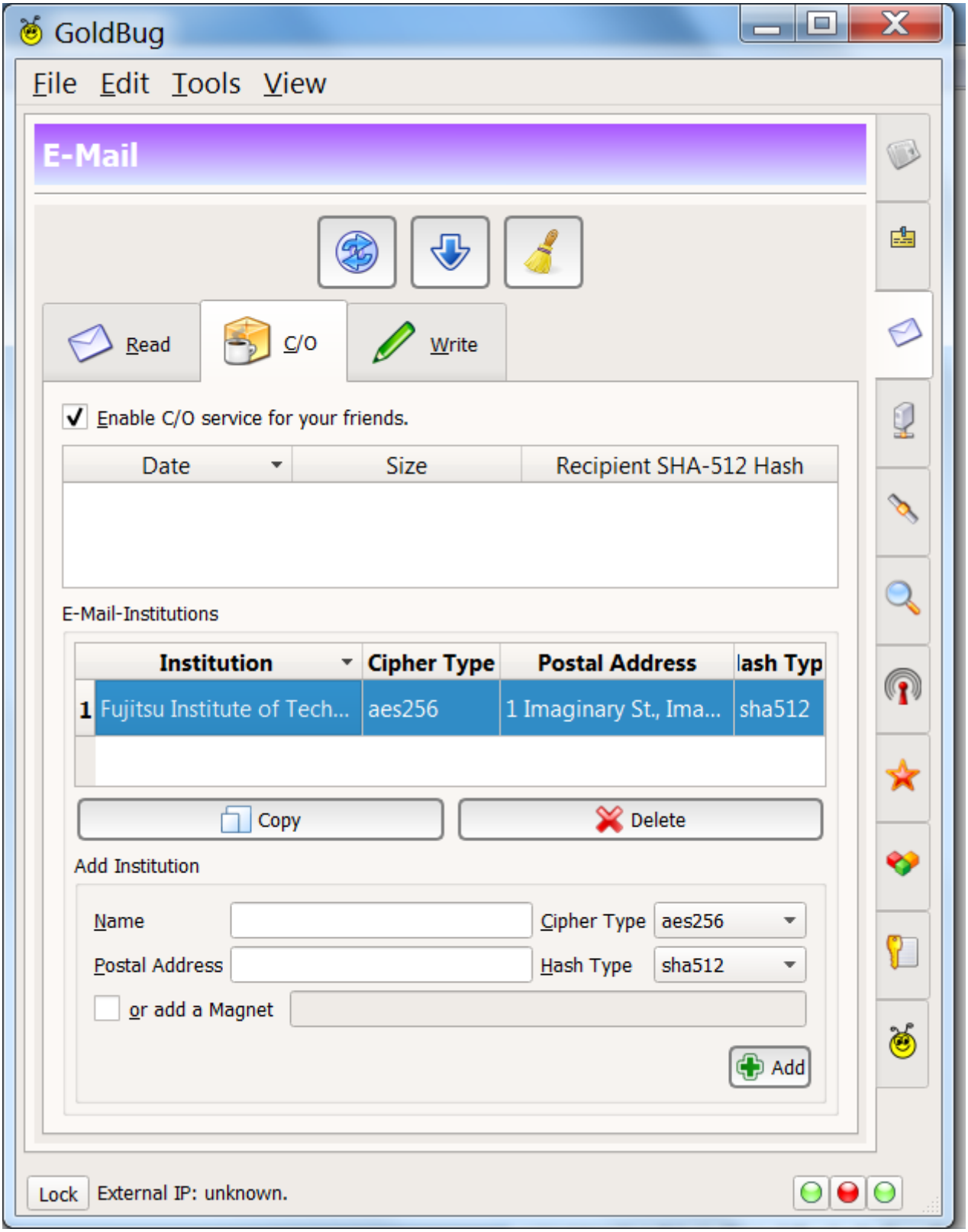
The interesting thing about the GoldBug e-mail feature - and here it may differ from other p2p e-mail implementations - is that it's also possible to send e-mail to friends who are offline.

There are two different methods for doing this:

9.4.1 Care-Of Method (c/o)

One method is to use a third, common friend to temporarily store the emails there with him. So, if Alice and Bob set up a common chat server on the web on their web server, and all three of them have swapped their keys, the web server acts like an e-mail inbox, as we know it from POP3 or IMAP.

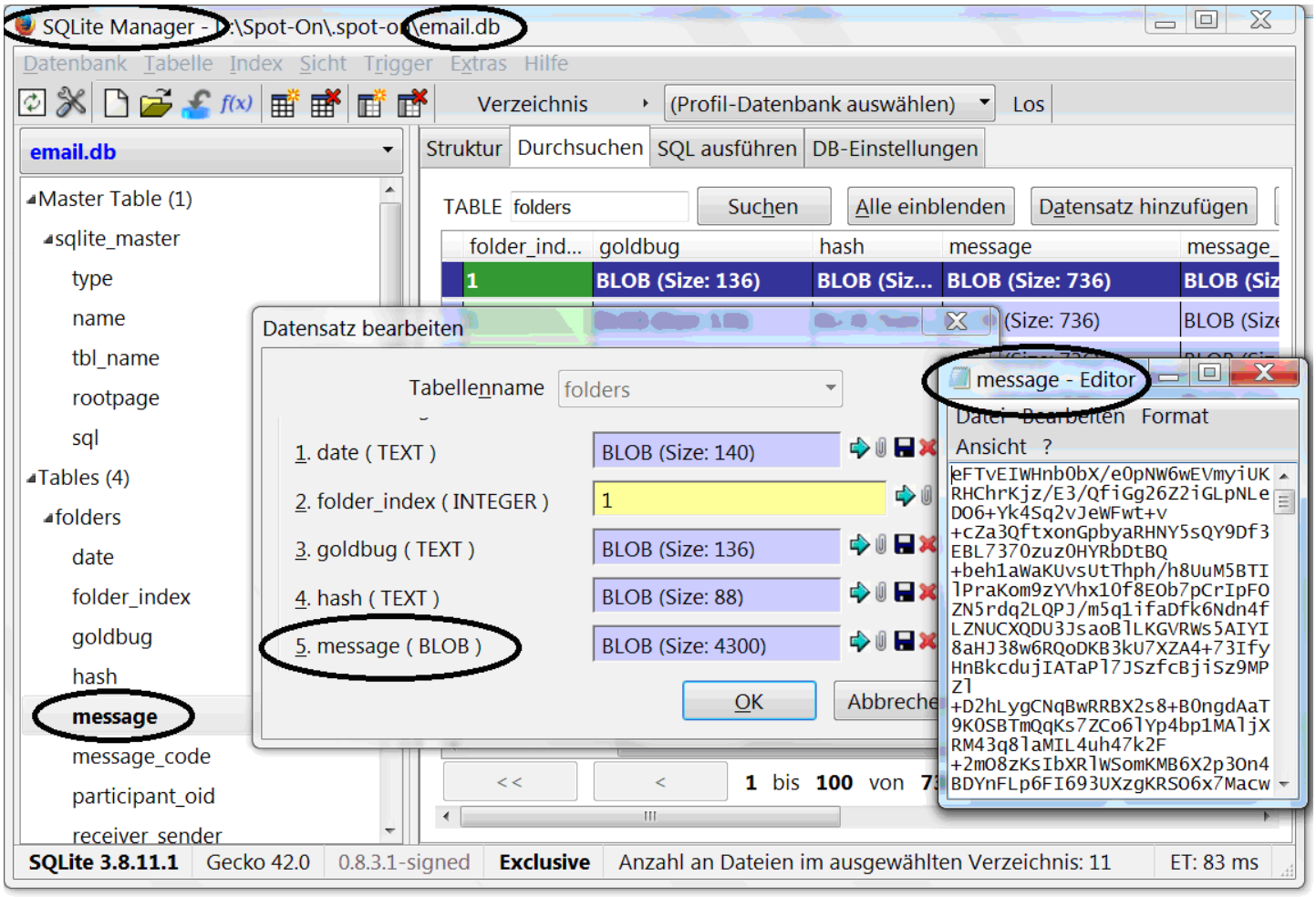
Figure: P2P e-mail from the postbox to a friend: c/o function



Basically, the e-mails do not need central servers; it can also be a third friend or a small Raspberry Pi computer at home, which remains online. It therefore makes sense to have more than one friend in your list and to network friends with other friends who can act for a caching. Since all e-mails are encrypted, the friends who provide a cache function cannot read the user's e-mail either.

Also, the emails are stored in encrypted databases. The figure shows that even ciphertext is displayed even when viewing the structure of the database file.

Figure: Database encryption



In order to activate this Care-Of (c/o) caching function, the check-box “Care-Of” must be activated in the sub-tab “E-mail Settings”. If two friends are then connected to each other and the third friend and want to enable the caching of e-mails in their own clients, then all have to insert the other two friends in the own e-mail contact list.

The GoldBug user can also choose to have the emails sent authenticated or unauthenticated in the p2p email, so they can simply be sent encrypted without evidence that the key belongs to a particular user.

The Care-of-P2P e-mail feature is one of the simplest in the software landscape for P2P e-mail at all. If three users share a common echo server and have added each other as a friend, only the C/O feature needs to be activated, and the e-mails are stored in the Friends of Friends cache, in case they are offline. Nothing is simpler than this architecture: you only need a few friends who want to participate in this process for internal communication within a group.

The second method is the establishment of a virtual e-mail institution. This is great for people who like to equip an entire community of friends with an email inbox.

9.4.2 Virtual E-Mail Institution (“VEMI”) method

For this it is also necessary to activate the C/O function with the check box as described above.

Then the user can create a virtual e-mail institution.

For the input and definition fields “Name” and “Address” of the institution, the user can freely get creative and choose any name. Then the public e-mail keys of the friends who want to save in this institution are still to be copied into this node.

Finally, the user can then copy out the created Magnet-URI-link and make it available to friends who then temporarily store in that mailbox. (For the Magnet-URI standard and what that is, see also below in the file transfer section with “StarBeam”). In addition, the node that sets up the e-mail institution must always also add the public e-mail key of the one for which it is to save.

The advantage over the first method is that the public e-mail key of the node setting up the institution need not be disclosed to anyone. With the c/o method, however, the public e-mail key has to be exchanged. Therefore, one can easily say that in the small Friends network a common node with the c/o function is ideal and the VEMI method of setting up virtual E-mail institutions tends to focus on vendors who want to set up mailboxes for a larger number of subscribers.

Settings example:

Here is an example of how the c/o function and the VEMI function, i.e. the creation of a virtual e-mail institution, are implemented step by step:

- The user activates the c/o function in the e-mail settings tab.
- The user creates an institution and chooses a name and address for the institution.
- Example: Name = “GB mailbox” and address = “Dotcom”
- The user inserts the email key of a friend into the own client. He then copies the available e-mail magnet from his e-mail institution and has the friends to paste it into their program. The magnet will look similar to this one:

magnet: in = GB mailbox & ct = aes256 & pa = Dotcom & ht = sha512 & xt = urn: institution

The user recognizes an e-mail magnet at his ending: URN = institution. Then you know that the magnet is not a buzz-group chat magnet nor a StarBeam magnet for file sharing - they would have the suffix “URN = buzz” or “URN = starbeam”. So then the own node will cache the emails of the friends in the established institution - also for addresses, which should be offline if necessary.

The user (as creator of an e-mail institution) does not need to exchange his own e-mail key with the friends or “subscribers” of his institution. The user can also exchange the e-mail keys of the friends in a group chat room via e’IRC/Buzz with the creator of an e-mail institution. The exchange process of key & email magnet does not have to impart any further identities.

9.5 Additional Encryption: Put a “GoldBug” on an e-mail

Regardless of which transmission method a user chooses, whether POP3, IMAP or P2P, the e-mails are always encrypted using the public (asymmetric) e-mail key and the echo protocol for transmission.

This is also the case if they are cached in an intermediate station such as a provider mailbox or a virtual institution or an intermediate node of a friend. Transport encryption and end-to-end encryption are consistent throughout.

As additional security for the e-mail function there is - similar to the so-called “Gemini” in chat-, now for e-mails the option to set a password on the e-mail: Not only the software is called GoldBug, but also the function in the e-mail client to set an additional password on the e-mail is called “GoldBug”.

Emails that have a “GoldBug” password set (see below the description of the file transfer function “StarBeam”, here the additional password is “Nova”) can only be read by the recipient if they have the appropriate “GoldBug “- so the user needs to know the golden key as a password. The user should therefore inform his friends about the password to be entered if the user sends them e-mails that still require an additional password in order to be opened.

This can be, for example, in the e-mails to your own wife, that the user always encrypts the e-mails with the name of the city in which the wedding or the wedding holiday took place.

Again, as with the chat with the Gemini, it is an important feature of symmetric and end-to-end encryption that the user can individually and manually create the end-to-end encrypting password.

In addition to the reminiscence of the short story by Edgar Allen Poe about a cryptogram and his work for cryptography in the early years of the beginning of industrialization - the GoldBug on an e-mail is a new idea, in addition to automatically encrypted e-mail created by the key exchange. It is asymmetrically encrypted e-mail, but also with a symmetric encryption - the GoldBug on an e-mail - even further encrypted in another layer per single e-mail, as this process, to touch each individual e-mail, is so far the standard (elsewhere by PGP, but symmetrical).

This process is done without additional encryption software, which elsewhere must be additionally installed e.g. as a plugin.

9.6 Forward Secrecy

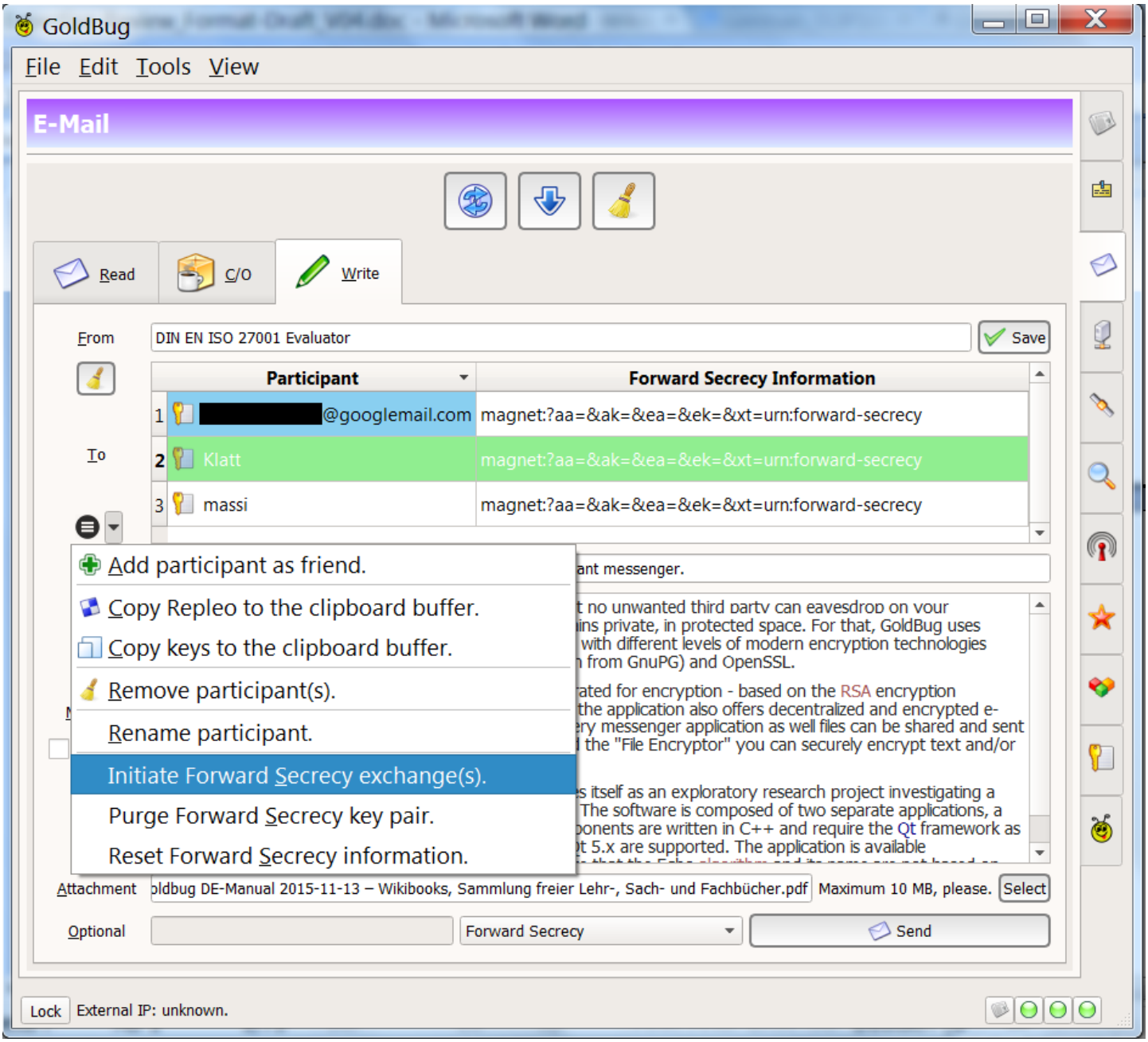
Using the included architecture of the Spot-on kernel, GoldBug is one of the first e-mail programs in the world to offer Forward Secrecy encryption, which can be both, asymmetric and symmetric for e-mail – so both methods within one e-mail Program are supported.

Forward Secrecy means – just to remember - that temporary keys are used to transmit the end-to-end encrypting password, so that if later on analysis should be made in regard of the

communication and the encryption, not the regular (permanent) key for the Communication is affected.

The user now sends his e-mail partner a session-based, symmetric (forward secrecy) key via the usual asymmetrical encryption of the e-mail key (see figure).

Figure: E-mail with forward secrecy



If the partner confirms the request and returns his temporary key, then both parties can use session-based asymmetric keys to further secure the email communication. Incidentally, this method of asymmetrical end-to-end backup was not only integrated for e-mail, but also for the chat function (see above: Forward Secrecy (FS) Calling).

The permanent public key is then used only to transport the session-based keys - not to transport the message (or: the previous message becomes the new key to the following message). That is, the ephemeral (temporary) key is shared with the partner via the permanent

public key. Then, if the ephemeral public key was correctly accepted by a recipient, said recipient also generates an ephemeral session key (symmetric), which is then sent back to the user via the user's public key as well.

The initiator then deletes its asymmetric ephemeral keys as soon as the temporary session has ended.

Figure: Email Tab: Forward Secrecy in GoldBug Email Client

So, when a user writes an email, GoldBug has four forward-secrecy modes available to encrypt the email:

- Normal encrypted: The email is sent as usual within the encrypted system (Echo or POPTASTIC), that is, the regular permanent symmetric email key is used to encrypt the message.
- Forward Secrecy Encrypted: Regular encryption uses session-based forward secrecy keys - that is, the user sends session-based keys over the permanent e-mail key channel and then encrypts his message with the temporary keys. So this adds to the message another asymmetrically encrypted level to the already existing email encryption.
- Pure Forward Secrecy Encrypted ("Pure FS"): The message is encrypted and sent only through the user's session-based (ephemeral) e-mail key. The permanent e-mail key is thus not used in the "Pure FS": This can therefore also be called the "instant" option within the e-mail process, that means it is immediate (in the sense of volatile) and a kind of one-time e-mail. This generates mail-addresses and mail-boxes - which can be deleted after the session. This creates one-time email accounts thanks to Pure Forward Secrecy.
- GoldBug encrypted: A GoldBug sets as described above a password on the e-mail (e.g. with an AES, symmetric encryption) and the user must inform his e-mail partner about the password, ideally verbally. The thus symmetrically encrypted message is then also sent via the asymmetric e-mail encryption (permanent e-mail key).

If the user selects the checkbox option "Plain" next to the e-mail text, the e-mail is not written in HTML rich text mode, but in plain text mode. Word plain text has nothing to do here in terms of an antonym to ciphertext.

Again with the following attention to understanding: through the permanent (asymmetric) key (for e-mail (or so in chat) ephemeral keys (as asymmetrical keys) are exchanged, which are then the basis for the use of end-to-end encryption. That means, the ephemeral keys can be deleted at any time after use and the communication is not tied to the identities.

One should not be confused here, because even the end-to-end encrypting symmetric passphrases are actually ephemeral keys. But it becomes more apparent if only the asymmetric temporary keys which are pushed through the permanent asymmetric e-mail keys are initially

referred to as ephemeral keys (so that this is not confusing to those who are dealing with the forward-secrecy process or the word “ephemeral key” for the first time).

The encryption levels in Forward Secrecy in the GoldBug e-mail program can be described as simplified as follows:

- External encryption level: SSL/TSL connection
- Possible, additional encryption level: permanent asymmetric e-mail key (not with “Pure FS” - otherwise: first-ephemeral-then-permanent)
- Further level, which may later be deleted: Ephemeral, temporary asymmetric key (used only to transfer symmetric keys)
- First encryption level via Forward Secrecy: Symmetric key
- Alternative first encryption level via a GoldBug: Symmetric key via a manually defined GoldBug on the e-mail. The message format is thus (TLS/SSL (AES-GoldBug (e-mail message)).) According to Encrypt-then-Mac, this can be called “GoldBug-then-Permanent.” The GoldBug on an email encrypts the text in the envelope.

Temporary keys are not derived from permanent keys and have no relation to them in the generation. Session periods are defined manually by the user. This means that unlike other programs, the session is automatically defined by the online coming and going back offline, but the user himself determines when he wants to use new session-based keys. Again this can be anytime and “instant”. (See above: IPFS).

The process or protocol for forward secrecy can be described as follows with this example:

1. I send you my postal address. This is public.
2. You also send me your postal address. This is public.
3. Our addresses are permanent. These addresses change only when we move.

- Days later -
    1. I make a unique envelope, an ephemeral envelope.
    2. I send you, and only you, my unique envelope. Of course, I use your postal address to send you this. We assume, only you can read the written sentences. I could also sign the draft with my signature.
- On the same day -
    1. You will receive my unique envelope and you will also verify it by my signature, if you like.
    2. You create a special letter.
    3. You bundle the special letter into the unique envelope I sent you.
    4. Once you have sealed it, only I can open it.
    5. You send the unique envelope back to my postal address. Optionally you can of course also sign the created bundle again.
- Still the same day -
    1. I receive your bundle. In this bundle is my unique envelope.
    2. In my unique envelope that only I can open is your special letter.

3. We use the special letter as often as we want ... Once, twice. Etc.

A set of session-based keys is sent back via the ephemeral key. The first bundle is transported via the permanent keys. Permanent bowls do not have to be, but they do exist (because the SSL/TLS connection is still there). That is, the user sends the ephemeral key (one way) over the permanent key, and the partner returns the set of session-based (symmetric keys) via the ephemeral key.

At the end - after the log is completed - the ephemeral keys are deleted and only the set of session-based keys remains.

9.7 Secret Streams

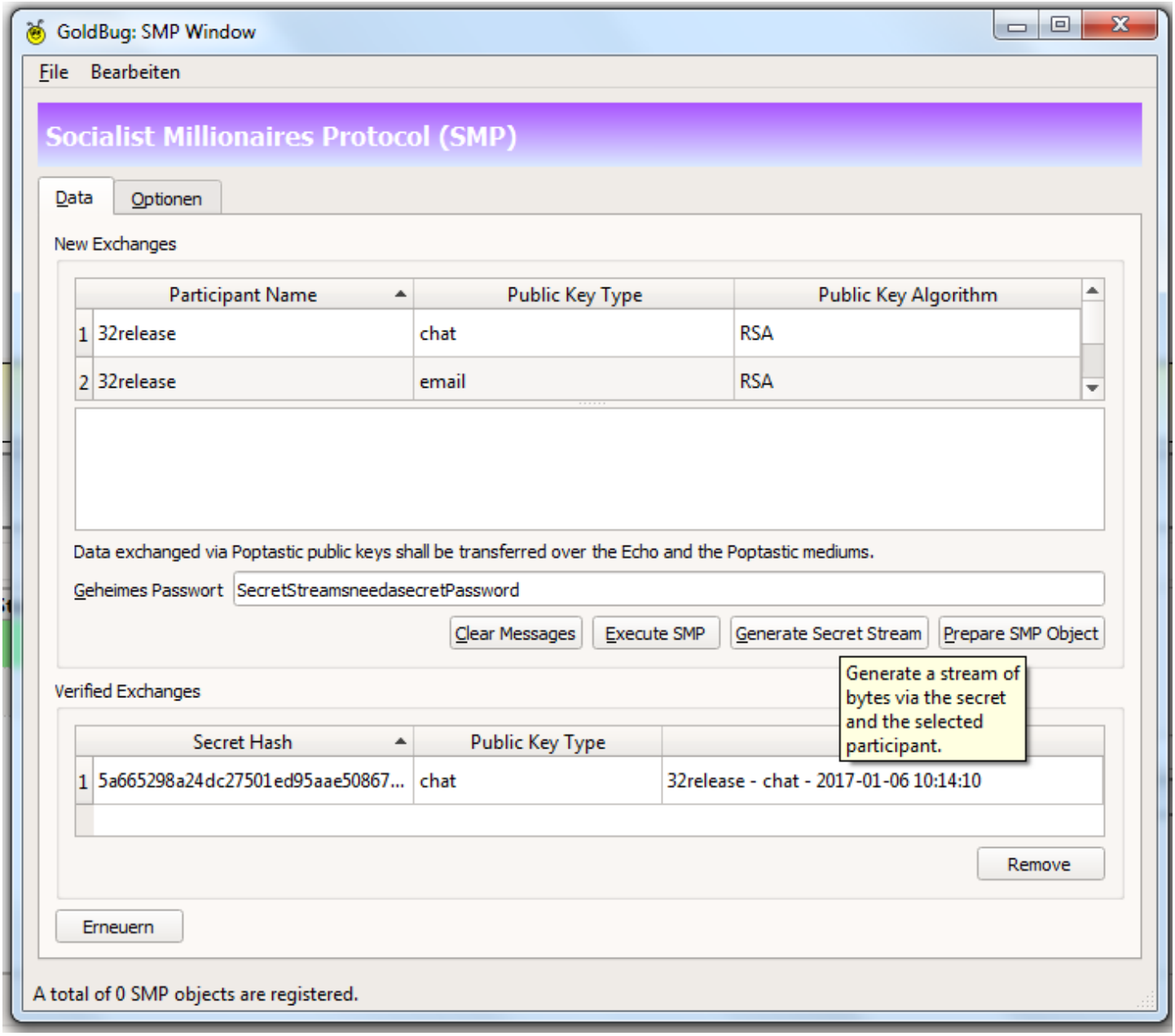
It has already been described as innovative in an e-mail client, offering both asymmetric and symmetric forward secrecy. The new and so far uniquely implemented function of the Secret Streams can be further appreciated as even more innovative: Secret Streams are, so to speak, a list of temporary keys generated by the password in the SMP authentication in the Socialist Millionaire Protocol. The SMP process has been extensively described above in the chat section and can also be used for cryptographic calling.

And now, this breathtaking new feature of the Secret Streams has not seen the world like this yet, is satisfying users and getting used to companions within the market - if that phrase is allowed - because it solves the key transfer problem fundamentally: both users receive a password known only to them in the SMP process through reciprocal contextual cues or just on commonly known secret. Once this authentication has taken place, this password can also be used to derive numerous temporary, ephemeral keys that are the same in both clients, without them having to be transmitted - because SMP authentication is the responsibility of a so-called zero-knowledge process.

The purpose of the SMP filter is to generate key streams from a secret. The secret is mathematically negotiated through the SMP process without it being transmitted as such. Thus, the keys of the secret stream function are derived from a zero-knowledge proof!

The function of the Secret Streams is available for chat, e-mail and also POPTASTIC: Temporary keys, which do not have to be transmitted anymore! Secret Streams should represent a small revolution in cryptography, because the password transmission problem would be partially solved. Only the SMP secret that was previously used for authentication, not yet for encryption, is required.

Figure: Implementation of secret streams in the extended SMP protocol



Further research perspectives

- It should be remembered that one transforms the permanent keys into transport keys. If these are compromised, the encryption becomes recognizable with the other encryption authorities. This concept creates a creative research area within the echo protocol environment. Here are some concepts suggestions that could be further incorporated:

- Participants could consistently generate ephemeral (asymmetric) key pairs and exchange session-based (symmetric) keys over the ephemeral keys. Attendees would be notified if there were not enough keys left. Replacement (from ephemeral keys to permanent key or session-based (symmetric)keys to (session-based) ephemeral (asymmetric) keys would then be automatically regulated ... similar to exchanging status messages via online status in chat exchanged only over session-based keys in the Echo or POPTASTIC protocol.

- Instead of exchanging one set of private session keys, multiple sets of private session bowls could be exchanged for supplies. Retaining data differently for a variety of anonymous email addresses with session-based keys.

- The OTR concept (so far for chat) could be applied within the permanent keys and also for e-mail. POPTASTIC in a different way, if chat goes via e-mail, then the chat key with OTR can also be sent via e-mail.

By using unique keys, information transfers in a session can be ideally protected - even if there are attempts to compromise.

That means, forward secrecy offers a substantial improvement in the protection of encrypted transmissions for little effort and no cost.

After describing e-mail and its numerous options for improved and innovated encryption, we come to the already announced term POPTASTIC - the function of chat over e-mail servers.

10 POPTASTIC - Encrypted chat and email via POP3 & IMAP

POPTASTIC is an innovation in messaging - encrypted chat over email servers.

With the POPTASTIC function all e-mail accounts, e.g. from Gmail, Outlook or Yahoo! -Mail can be encrypted asymmetrically end-to-end with GoldBug - and additionally hybrid symmetrically. The clou: every POP3 or IMAP server can now also be used for encrypted chat. And that also through firewalls when e-mail is given and going out.

Figure: POPTASTIC

Let’s take a closer look at POPTASTIC here at desktop client GoldBug.

10.1 Chat over POPTASTIC

So why should you still use a dedicated chat server or secure chat protocols with plug-ins for encryption, if you can just use the own e-mail address for e-mail and chat at the same time? The multi-decade old POP3 protocol and numerous email servers can now be used for encrypted chat with GoldBug. The e-mail server is simply converted as a chat server.

For this, the chat message is converted into an encrypted email, sent via POP3 or IMAP, and the recipient is converting it back into a chat message. Since the GoldBug Messenger is also an e-mail client at the same time, the encrypted message exchange also works via e-mail. The program will automatically detect if it is an email via POP3 or a chat message.

Chat and email through POPTASTIC are proxy enabled and can therefore be operated from work, the university or behind a firewall, even through the Tor network. If you log in to your e-mail account with a web browser, you can see what the encrypted message looks like.

The additional symmetrical end-to-end encryption via POP3 can - as with the echo protocol - not only be used as forward secrecy, but can also be renewed “instantaneously” every second. Therefore, here too (as above) of Instant Perfect Forward Secrecy (IPFS) is spoken, which is now possible via POP3 and IMAP for the POPTASTIC chat! Finally, there is also the option in POPTASTIC of making a call for the transmission of a Gemini using the methods differentiated above.

This option variety of the chat encryption with POPTASTIC is not given so far also with the architectural derivatives for mobile devices.

Anyway, for users surely an interesting and easy way to chat encrypted via this email protocol.

10.2 E-mail via POPTASTIC

Just as there is e-mail utilizing the e-mail key and as to chat over the POPTASTIC key, it is also possible to e-mail via POPTASTIC. Since POPTASTIC is a key which the friend is adding to the own client (via the friend-add tab), the POPTASTIC contact or the e-mail address is provided with a lock symbol and additionally marked with a background color to indicate that the message exchange here always happens only encrypted.

If you add an e-mail address in the add-friend tab, that contact will also be added to the contact list in the e-mail - but without the locked icon and background color. This indicates that the e-mail messages are unencrypted with this contact. This is the case if someone does not use the GoldBug client. Then POPTASTIC will send the email unencrypted to the @mail address.

The program knows: if you mail from the POPTASTIC key to a POPTASITC key, then this is always encrypted and can also be chat. And if you mail from the POPTASTIC key to an @mail address, then the message is unencrypted. This is the only and rare case that the client leaves the

message unencrypted, since it does not use the echo protocol, but the regular e-mail protocol SMTP.

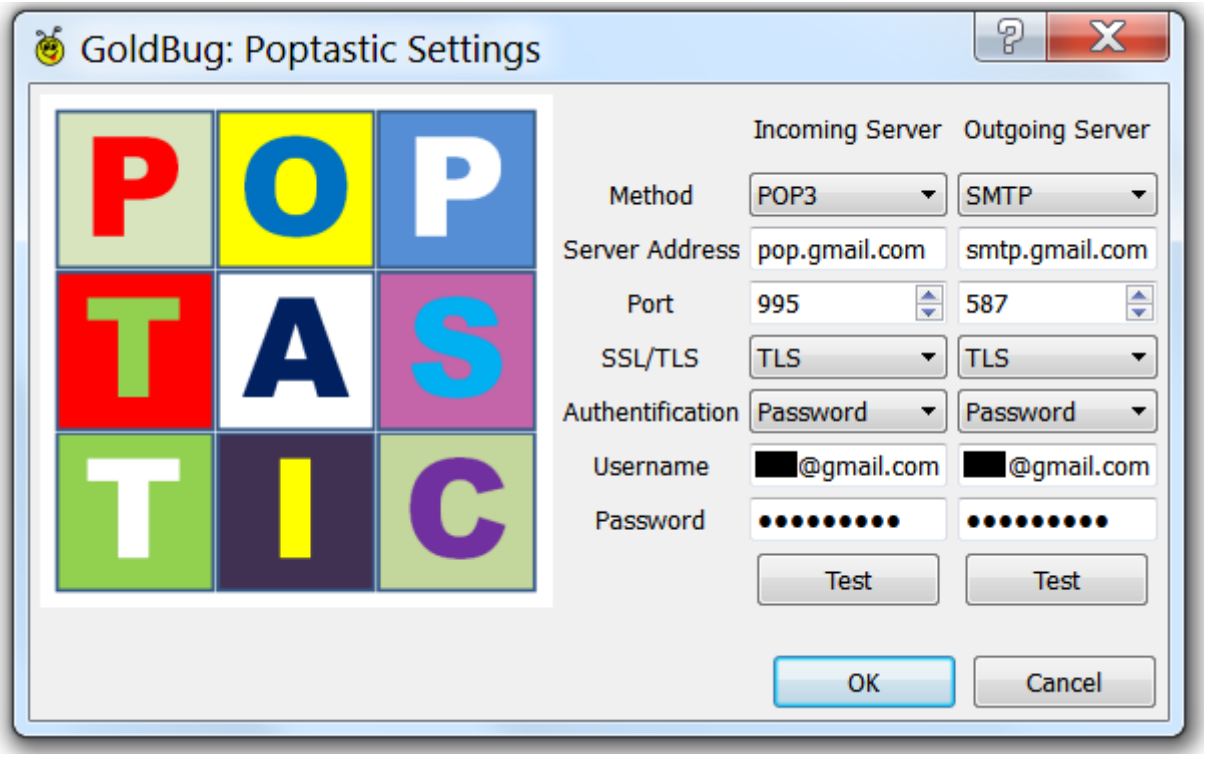
However, if the contact also uses GoldBug, both can permanently email encrypted when the POPTASTIC key is entered in the Add Friend tab.

E-mail via POPTASTIC is then a simple permanent encrypted e-mailing, by simply swapping once the POPTASTIC key at the beginning.

10.3 Setting up POPTASTIC

A detailed description of the configuration options of the e-mail server can be found above in the section on POP3 and IMAP (see also figure).

Figure: POPTASTIC Settings: Encrypted Chat and Encrypted Email over POP3 and IMAP



Note: In Gmail you should set the option on the Web that retrieved POP3 messages are deleted from the INBOX. To connect, it's also a good idea to set the security setting in Gmail so that you can connect to all your local email clients (Gmail should allow unknown clients):

- Settings / Forward and POP & IMAP / POP Download: Enable POP for all mail
- Settings / Accounts & Import / Change Account Settings: Other Settings / [New window] / Security / Access for less secure / unknown Apps: Enabled.

It may be advisable to set up an extra e-mail account for a first test and further use: It may be important to note that new e-mail accounts, e.g. for Gmail, may be limited to the first 30 days for the sending of e-mails (e.g. Gmail for 500 chat messages or e-mails per day). This should be sufficient for a test or normal need if necessary. Otherwise you can set up your own e-mail

server with GoldBug and you are no longer dependent on the @Mail accounts of the major providers, if it is a small internal user groups that use the echo mail - represented here with its own server need.

10.4 Further development of POPTASTIC

This idea of the POPTASTIC architecture has been developed by the GoldBug development team, published and described also in the study Big 7 by the auditors of the program. Then this idea has been taken over by the mobile application Delta-Chat, although the encryption there also runs via PGP and exclusively only via IMAP server. The original commits and publication data show the historical origins to which references and credits should be made in order not to seek the proximity of any meaning of plagiarizing the architecture of this encrypted communication.

A fork and a progression of the POPTASTIC idea that is to be welcomed (and referenced) is in a mobile chat client with an appealing user interface. (See the release of POPTASTIC in 2014 and further publication in mid-2016 and its derivatives with first commits a few years later: https://sf.net/projects/goldbug/files/bigseven-crypto-audit.pdf (p134, 2016), https://sourceforge.net/p/goldbug/wiki/release-history/ (2014), https://delta.chat/en/blog (2016)).

11 FileSharing: with StarBeam

As in any messenger, a file transfer in GoldBug is also possible and in general this FileSharing function is always encrypted between two defined friends or even multiple people. This happens in the tab “StarBeam”. The term StarBeam (SB) implies that FileSharing should be as simple as the light of the stars projected or “beamed” through the galaxies.

While traditional file-sharing programs such as EMule or BitTorrent have initially relied on specific links such as the ed2k link or the torrent link, file transfers now have to do with the linking of files using the Magnet URI standard, which is known from both, torrents and nearly all of the more advanced Gnutella clients, even for the Edonkey network it is established in the Shareaza client.

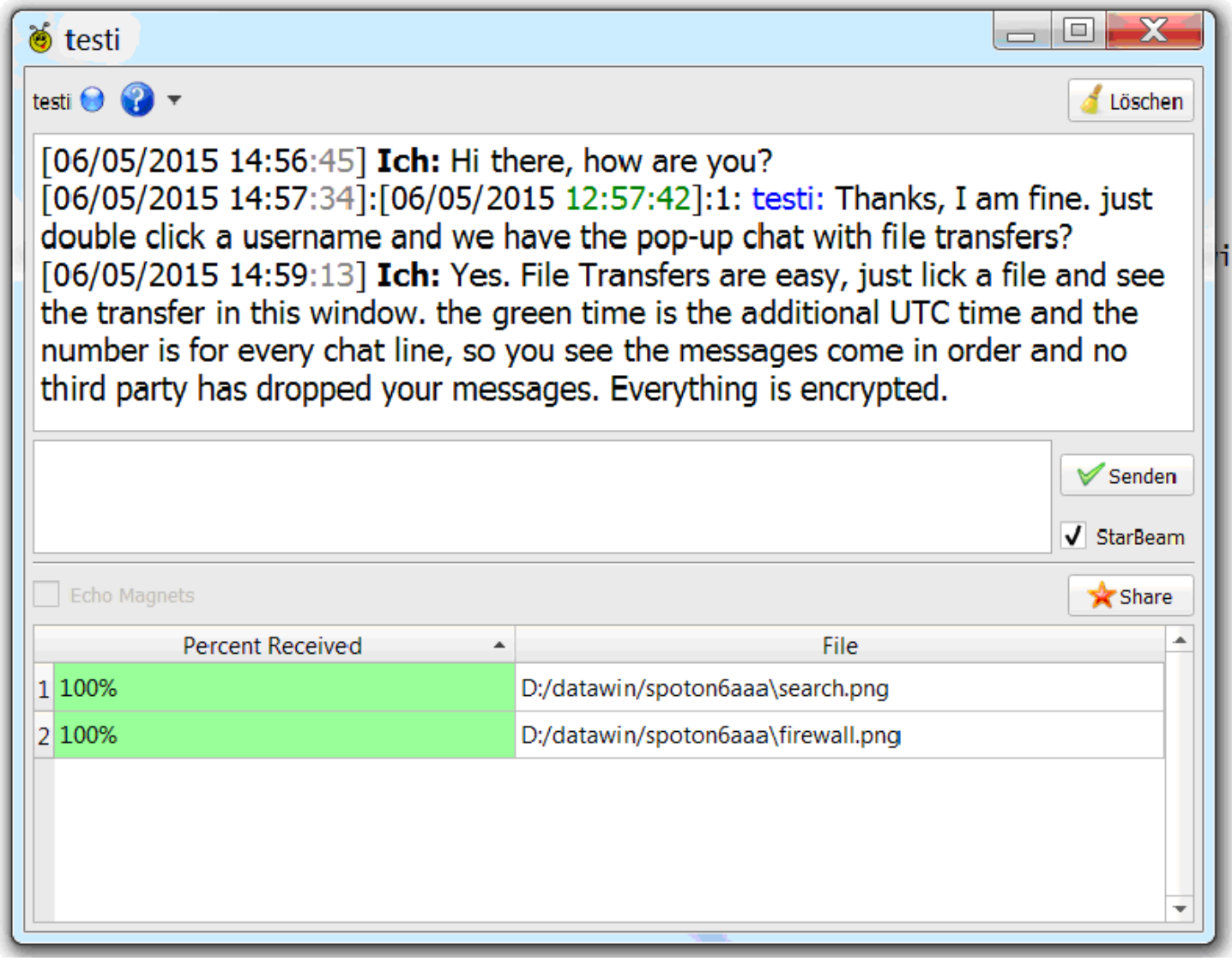
The elaboration of GoldBug and the Spot-On-kernel has developed the architecture of this Magnet URI standard further and added cryptographic values.

If the user now wants to download a file via GoldBug from others, the user has to copy a magnet URI link into the program. And accordingly: If the user wants to prepare an upload of a file, a Magnet-URI has to be created for this file.

This process is considered as simple as possible: If the user is chatting with a friend in a pop-up chat window (see illustration), there is a button “Share StarBeam”. The user can simply click this,

then select the file to be sent and it is already securely encrypted transmitted over the echo connection to the friend.

Figure: GoldBug messenger chat pop-up window with file transfer



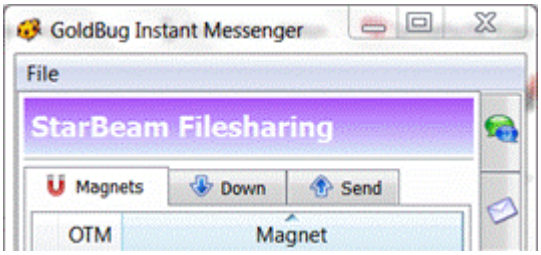
So the user can easily and securely transfer a ZIP with holiday pictures to family members via the chat or the StarBeam tab.

In order to send a file to an entire group, the user can also post his magnet in the group chat. This will then be automatically added to the downloads (see Checkbox in the menu Options: Buzz / e'IRC-Chat: accept magnets).

Due to the echo protocol, the individual packages are also "swarmed", i.e. the encrypted packets that pass by the user, are even shared with his friends and neighbors.

The File-Sharing StarBeam Tab consists of three sub-tabs: one for uploading, one for downloading, and one for creating or adding SB Magnets.

Figure: StarBeam with its three sub-tabs



11.1 Creating StarBeam magnets with encryption values

A Magnet-URI is a standard known from many file-sharing programs (many in the Gnutella network) or torrent links and also corresponds to eDonkey / Emule ed2k links.

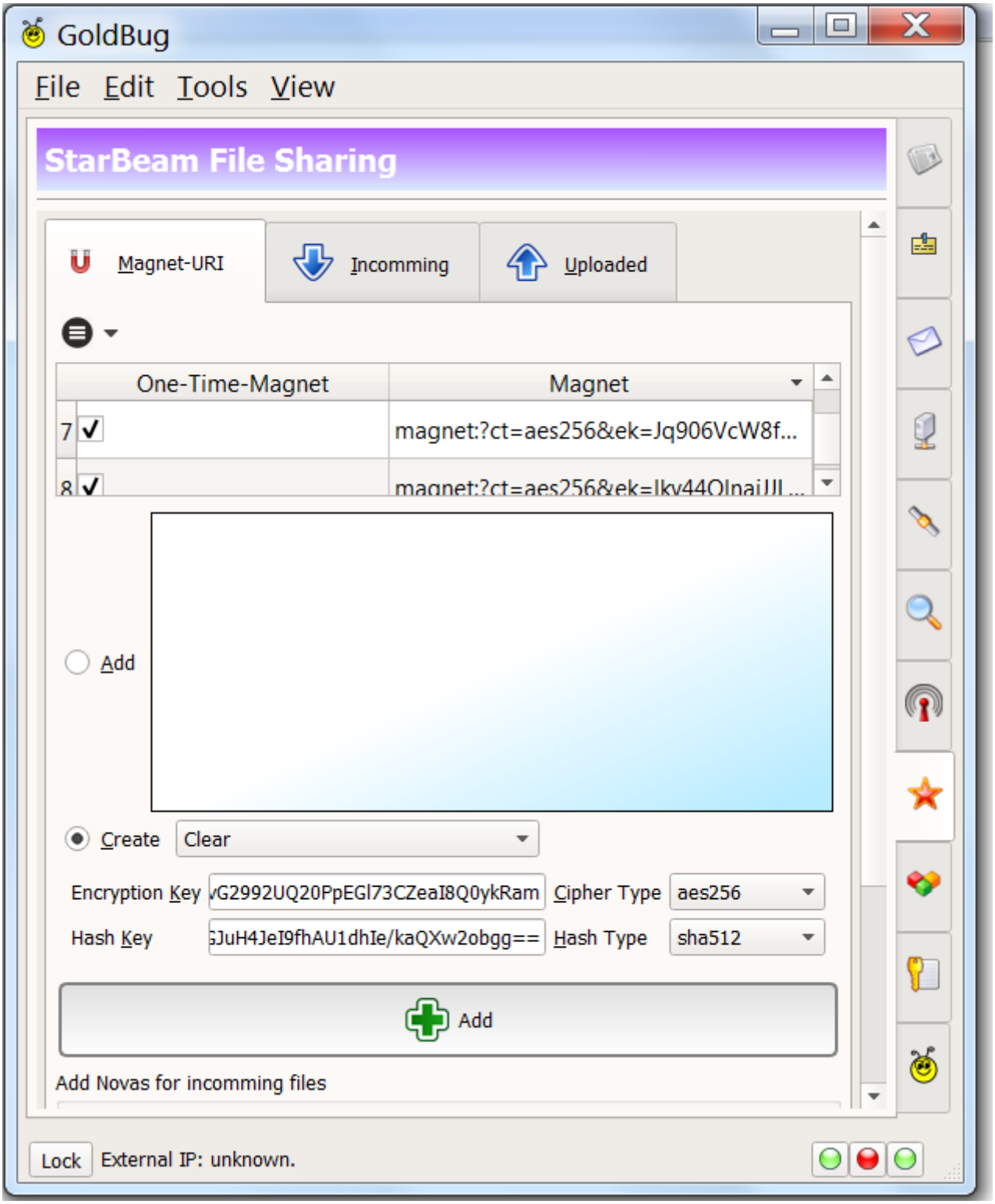
The further development of the Magnet-URI standard by the GoldBug Messenger underlying Spot-On library lies in the design of the Magnet URI with encryption values. Magnets are used to create or hold together a bundle of cryptological information.

Between the nodes in the echo network is thus created an end-to-end encrypted channel through which a file can then be sent. However, any further file can be sent as well. The Magnet is thus not associated with a particular file. The SB magnet is like a channel through which an instance can continuously and permanently send files - or there is a one-time magnet created, which is deleted immediately after the single use.

This dual-use effect does not allow a magnet to be associated with a single file or IP address. Also, a filename does not appear in the SB magnet (as is the case even with the also more advanced links, for example, from OFFSystem or RetroShare compared to Gnutella, Emule, and Torrent links). Thus, it becomes clear that no specific file is exchanged in StarBeam, but only encrypted channels are exchanged. A “wormhole”, so to speak, to stick to the concept of the “star trek” movie. And this channel is defined by a Magnet URI link and its cryptological values.

While many opinions see the linking of Gnutella, Edonkey, and Torrent links on the Web as critical, there is no reason to scrutinize those values in a collection of encryption values. A homepage or independent portal with StarBeam and the Magnet-URIs present an advanced concept. In addition to the conceptual choices of selecting a link standard, the usage aspect is also about the security of the file transfer between two private users.

Figure: Magnet URI standard with crypto values for file transfer



In summary: To send a file, an encrypted channel must be created. This works with the creation of a Magnet - marked at the end by URN=StarBeam.

Then the file is transmitted encrypted - packet by packet - over this channel using the HTTPS protocol (which can be based on TCP, UDP and also SCTP or even Bluetooth connections). It is therefore an interesting question for a practical test, whether a transfer of a large, encrypted file via StarBeam via the Echo protocol based on SCTP, TCP or UDP connections is ceteris paribus transmitted error-free and fastest?

For the process of private file transfer from friend to friend some more notes:

11.1.1 " 'Option "Nova": Encrypt the file before transferring the file?' "

Before the user sends a file, he can consider whether he simply attaches it to an e-mail within the GoldBug e-mail function. This is the variant of choice if the file is smaller than 10 MB. Larger files should only be transferred to a friend via the StarBeam feature or in the chat window.

Before transferring, the user may also consider encrypting the file on the hard disk. To do this, the GoldBug Messenger provides a tool for file encryption, found in the main menu under tools (see the section below: GoldBug-File-Encryptor). A double passphrase encodes the file in it.

Of course, this tool of the GoldBug File Encryptor can also be used if you want to upload a file somewhere to an online hoster of the cloud or transfer it via another path, another messenger or e-mail.

However, as these online hosting sites like Dropbox may control files and mark encrypted files with a question mark, even though it should be an exclamation mark, it makes sense to transfer the encrypted file from point to point, from friend to friend, directly via GoldBug and to use no external or foreign intermediate cache as a host.

Some pack the files in a zip and encrypt it before sending or uploading. However, Zip encryption is very easy to crack with 96 bits, so you should use a key recommended for RSA - today for RSA with at least 2048 bits, better 3072 bits (as defined for GoldBug as default). Or you can use better McEliece instead of RSA.

No matter how the user prepares and transfers the file: (1) as a plain binary file, or (2) encrypted with the GoldBug tool via StarBeam or (3) as a file with an additional Nova password (see below) as a protection method in the Star Beam process - in any case, it will in turn be encrypted several times using the Echo Protocol. An optimum of encryption and a variety of options, which cover numerous wishes.

Just like you can put an additional password on an e-mail (see above, called "GoldBug" in the e-mail function), you can also set another password on the file - respective on the used magnet-URI for the file transfer. This is called "Nova".

Even if the file transfer is successful or even a third unknown party could crack the previous multiple encryption (which is not to be assumed), the nova password introduces end-to-end encryption, which is secure as long as the shared password is exclusive to both partners.

Because, if the transmission of the SB magnet should be intercepted - the user somehow has to transfer the magnet online to his friend - then anyone who knows the magnet can also receive the file as well. Therefore, it makes sense to protect the file with a "nova" - a password that both friends have exchanged, possibly orally, in the past or via a second channel.

The Nova also builds on the end-to-end encryption standard AES (that means the password string is generated by the computer, if the user does not think up his own passphrase).

As mentioned, the ability to create your own end-to-end encrypting passwords yourself and manually enter them - known in the science as “Customer Supplied Encryption Keys” (#CEKS) - is so far implemented only in a very few applications such as GoldBug Messenger or Smoke Chat.

And: The Nova must have been deposited in the node of the recipient - before - the file transfer begins!

11.1.2 Using a one-time magnet

Ideally, the user has his own Magnet-URI for each file. That would then be a one-time magnet (OTM), a magnet that is used only once for a file. (OTM is the same as the idea of an OTP - a one-time pad: a string that is used only once.) OTP is often considered essential in cryptological processes to provide security.)

The user can also use a magnet permanently, then it is like a subscribed video channel in which, for example, a new file is sent every Monday.

This also opens up completely new possibilities for torrent portals, for example: there does not even have to be a web portal in which thousands of links are linked! The portal itself needs only a single magnet in the decentralized echo network, then consecutively, one by one, it is possible to send one file after the other through the wormhole.

As soon as the user has transferred a file via the magnet, the user can delete or retain the Magnet-URI. If the user creates the Magnet as an OTM and activates the checkbox for OTM, it deletes itself after file transfer. This is similar to the movie Mission Impossible or apps for pictures where messages and pictures destroy themselves - The magnet is, so to speak, a StarBeam wormhole that closes again after a single use.

11.1.3 Overview of Magnet-URI Standards for Cryptographic Values

The following overview explains the usual cryptological values in the Magnet-URI standard.

Abbreviation	Example	Description
Abbreviation	Example	Description
rn	&rn=Spot-On_Developer_Channel_Key	Raumname
xf	&xf=10000	Exact Frequency
xs	&xs=Spot-On_Developer_Channel_Salt	Exact Salt
ct	&ct=aes256	Cipher Type
hk	&hk=Spot-On_Developer_Channel_Hash_Key	Hash Key

Abbreviation	Example	Description
ht	&ht=sha512	Hash Type
xt=urn:buzz	&xt=urn:buzz	Magnet zum IRC Chat
xt=urn:starbeam	&xt=urn:starbeam	Magnet zum Dateiversand
xt=urn:institution	&xt=urn:institution	Magnet zum E-Mail-Postfach

This standard is used to exchange symmetric keys for group chat or e-mail institutions or even file transfers with StarBeam.

magnet:?rn=Spot-On_Developer_Channel_Key&xf=10000&xs=Spot-On_Developer_Channel_Salt&ct=aes256&hk=Spot-On_Developer_Channel_Hash_Key&ht=sha512&xt=urn:buzz

The Magnet-URI standard has been further developed into a format to pass on encryption values similar to a blood count sheet.

Encryption with very individual DNA-values provide the highest possible security.

11.1.4 Rewind function

If a recipient has received a file packet, a chunk (or in the spot-on kernel also called “link”), he is able to upload it again - even in other magnet URI channels. - Or it can be sent again into the same channel. This is similar to a rewind function: the file is simply played again via the echo network - like on a cassette recorder or MP3 player. The file can also be sent many hours or days later. Anyone who has received a copy via the Magnet URI channel becomes a satellite, and can re-import the data into the defined channel, or better, via the or a StarBeam magnet.

11.1.5 Comparison with turtle hopping

Turtle hopping will pass the file packages from friend to friend until they reach a defined destination. It is a transformation of a peer-to-peer (P2P) network into a friend-to-friend (F2F) network. However, it has the disadvantage that friends with little upload speed to the next friend in the chain form a bottleneck and slow down the transport:

The turtle-hopping protocol is firstly connected only to nodes that have been defined as friends and here in this chain of friends can be a friend, which performs only a small bandwidth. This then acts as a bottleneck and senders and recipients of the file must necessarily send through this bottleneck.

The transmission of a file in the StarBeam function via the echo protocol is therefore also more effective than using a protocol similar to “turtle hopping” (currently only implemented in the RetroShare program), because here, depending on the design of the echo network (full echo, half echo, adaptive echo) the basic encryption nodes with low bandwidth do not have to act as a bottleneck, they optimize the desired download speed via other echo paths.

When sending files via the echo protocol, therefore, other nodes such as peers or paths via other graph-options can be included in the hopping over intermediate stations if there is a faster route somewhere:

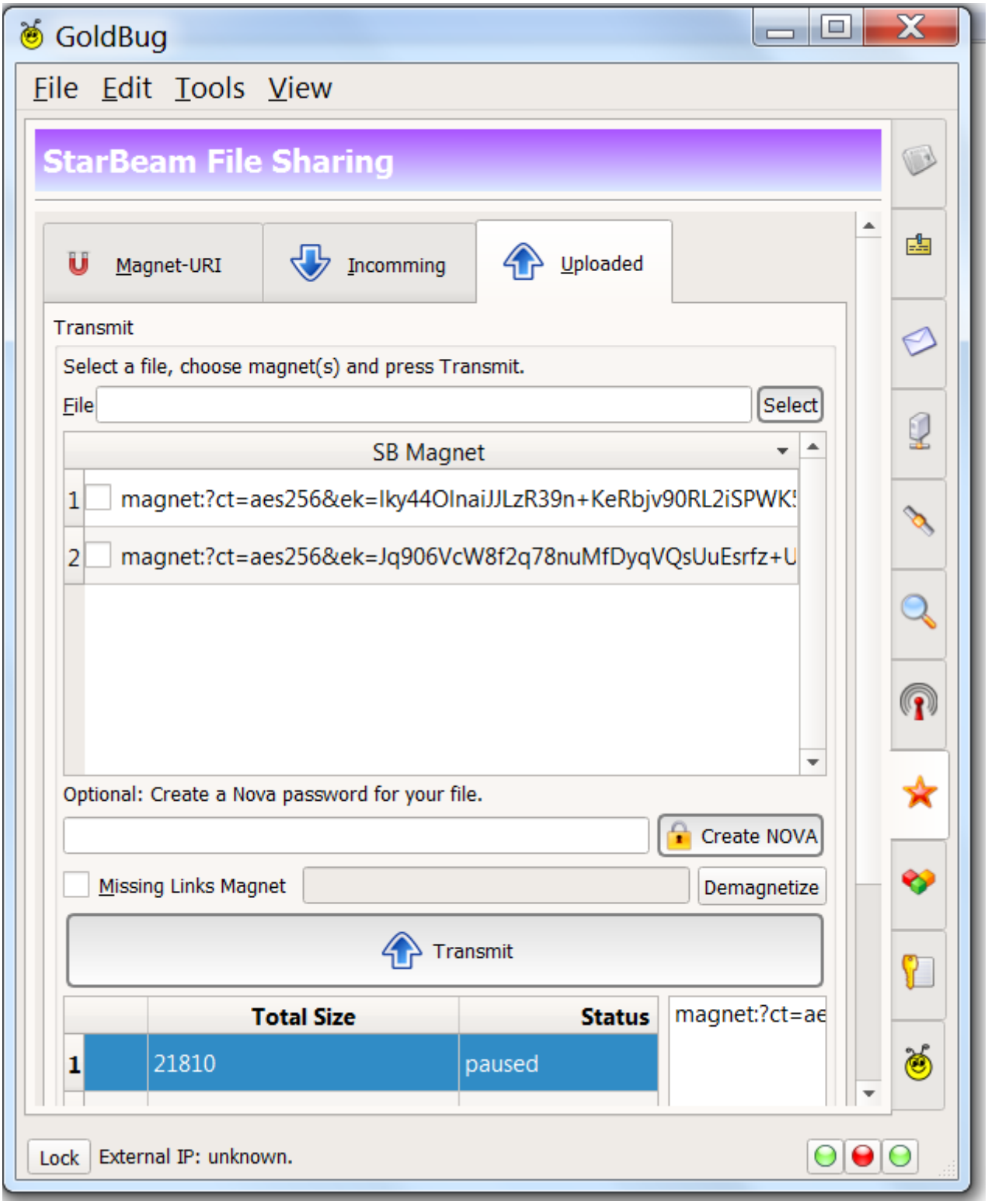
The echo protocol automatically creates the flow in the network of nodes (simply by allowing each node to send encrypted file packets to each linked node) and therefore also chooses the fastest path of all possible graphs to the desired node.

11.2 StarBeam upload: transfer a file

As described above, sending a file from the chat window to a single friend is very simple: with the Share-StarBeam button you just have to choose one file and it will be transferred to your friend.

In the following, we now look at the upload process with its technical details in the sub-tabulator “Uploads” of the StarBeam tabulator.

Figure: Starbeam file transfer: uploading files



If the user has defined and generated a Magnet URI, it will appear not only in the sub-tab for the magnets, but also in the table in the sub tab for the upload/seed.

Also from here the upload of a file can be started. To do this, the user selects with the check box a Magnet in this sub-tab for the upload. Likewise the file is selected.

Optional: A nova password for the additional encryption of the file: Finally, the user can still decide whether he wants to put on the transfer an additional password - as described above: a “Nova”. The friend can open the file only if he enters the Nova password. It is an additional symmetric encryption to secure a file transfer.

Then press the “Transmit” button.

(Tech Note: Since the echo is transmitted as HTTPS Post or HTTPS Get, the transfer is the same as a web page. The chunk size can be left as predefined, as it is in the minimum view GoldBug interface hidden. In case the pulse size is made larger, the web page being transferred becomes longer, so to speak.)

Finally, the user copies the Magnet URI and sends it to his friend. The user can copy the magnet URI via the context menu button.

If the friend has copied the magnet, the user can start the transfer by deactivating the pause function (check box “Pause” in the table).

Then the file is transferred to the friend.

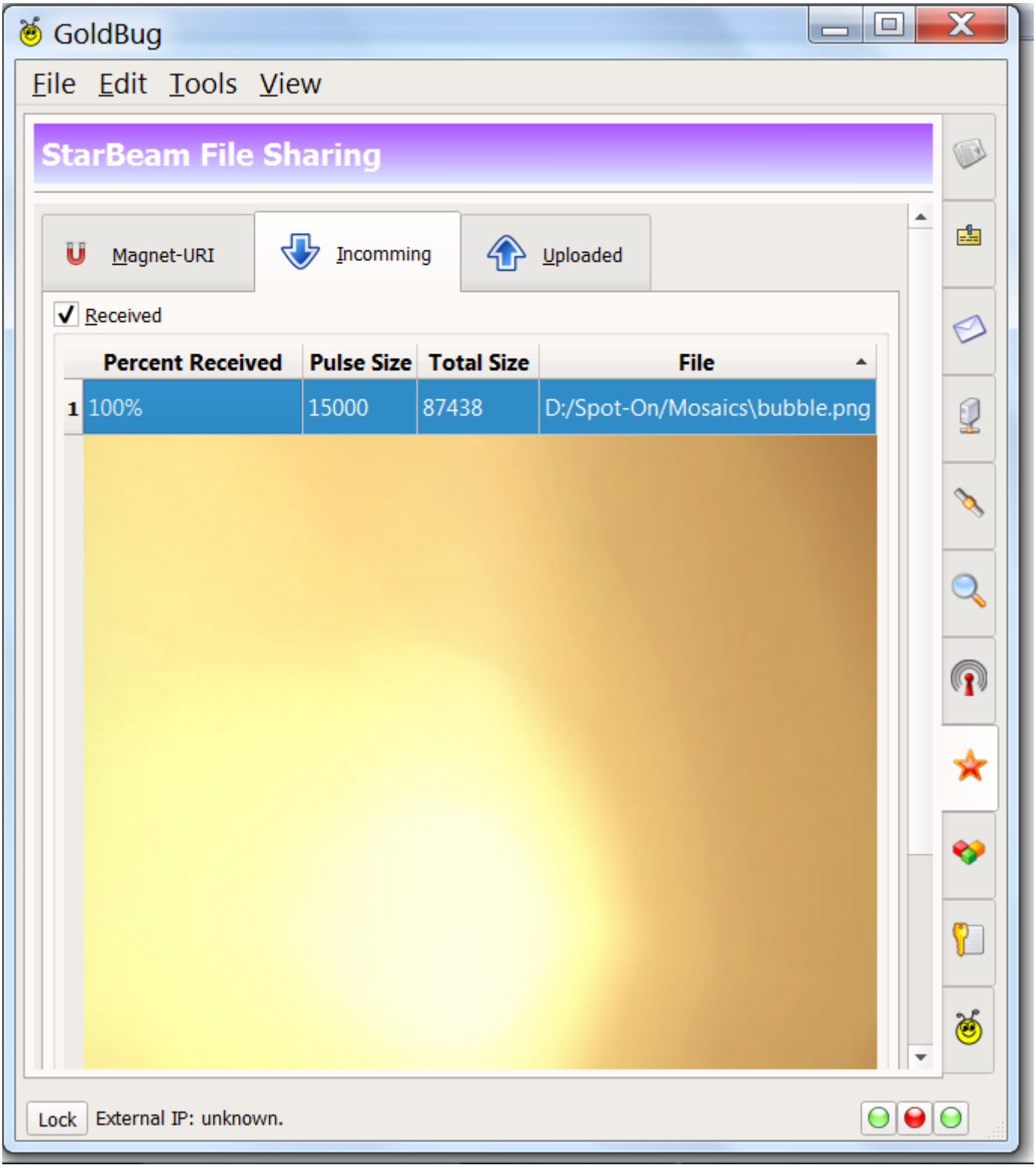
11.3 StarBeam downloads

To load a file with StarBeam, the user needs the StarBeam Magnet of the file. The user receives this from his friend, who wants to send a file.

The user then simply copies the magnetic URI into the sub-tab for the magnetic URIs. Before that, the user should activate the checkbox “Receiving” in the download sub-tab. This is deactivated in advance by default, so that no unwanted files are received.

The user then tells his friend that he has inserted the magnet URI and then the friend can start the transmission. The download starts as soon as a transmitter sends the file via the echo and through the crypto channel of the Magnet.

Figure: StarBeam File Transfer - Incoming Files



With the additional settings on this tab page for the upload, the user can still define the size and the path for the download area.

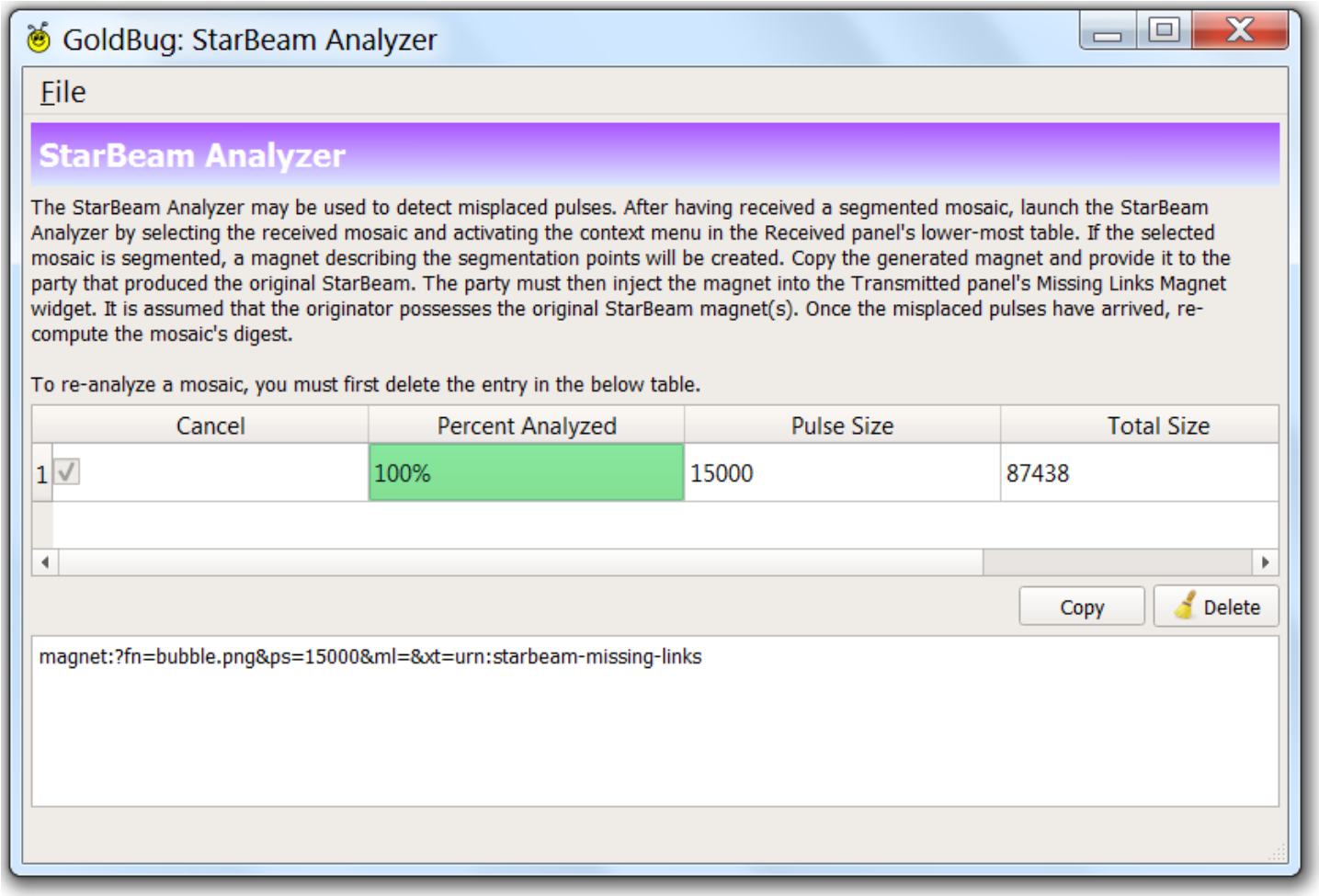
Successfully downloaded parts are called “Mosaics” within StarBeam and stored in the same path of the installation on the hard disk. Similar to a puzzle, the mosaic pieces are assembled into a full mosaic, the resulting file.

The still-to-be-transferred file parts are called “links” in StarBeam (see also the term “chunks” in the old EDonkey network or the term “blocks” in the Gnutella network, which was coined by the use of the then there used Tiger–Tree-Hashes).

11.3.1 Tool: Star Beam Analyzer

If a file was not successfully transferred 100%, it can be checked with the StarBeam Analyzer tool. This determines if all mosaic parts are present or if any links / chunks / blocks or packages still to be transferred are missing. If any links are missing, the SB Analyzer will create a magnet URI that the friend can re-enter in his upload tab. Then only the missing links or mosaics are sent again.

Figure: File transfer using Starbeam: analysis tool for the chunks



The file would also complete if the sender sends it three times a day over the echo with the “Rewind” function.

It should be noted that a magnet is a channel, and existing files in the local mosaic path will then be renewed if no one-time magnet is used and they are sent again in the same channel. A renewed shipment of the file by the uploader will thus overwrite the file received by the user again, if he has not set a lock option in the transfer table. The checkbox “Lock” does not delete the file that the user received.

11.3.2 Outlook for Crypto Torrents

Because of encryption, nobody can see what file a user is downloading, because nobody knows if the user was able to successfully decrypt the package - and even if, nobody knows if the user

created or saved the file in total from it.

The upload is similar. The upload is only visible from a neighbor IP, if this neighbor knows the Magnet of the file. In this case, if you want to load public Starbeam magnets, it is best to connect only to neighbors or chat servers that you trust or define as friend through account access.

Also, the above-mentioned variant of setting a Nova password on the file and the distribution of the physical blocks in time before granting the access rights to the nova password in a second process can offer new perspectives in technical, procedural or even legal considerations.

This means e.g. that the transfer of the file takes place in the past and the transfer of the decryption option takes place in a future, separate and downstream process.

Then, using the Echo protocol, StarBeam Magnet-URIs can play a role in new ways of thinking about developing and using the “crypto-torrents” discussed in the file-sharing community.

Encryption basically means that unauthorized persons do not know what is in the encrypted packet and that the owner of the key decides himself when to perform the decryption. That is, encryption has been logically applied to the file transfer and the sovereignty of the user.

12 Web search engine with URL database

With the integrated function of a web search GoldBug is also an open source p2p web search engine due to its used architecture of the kernel.

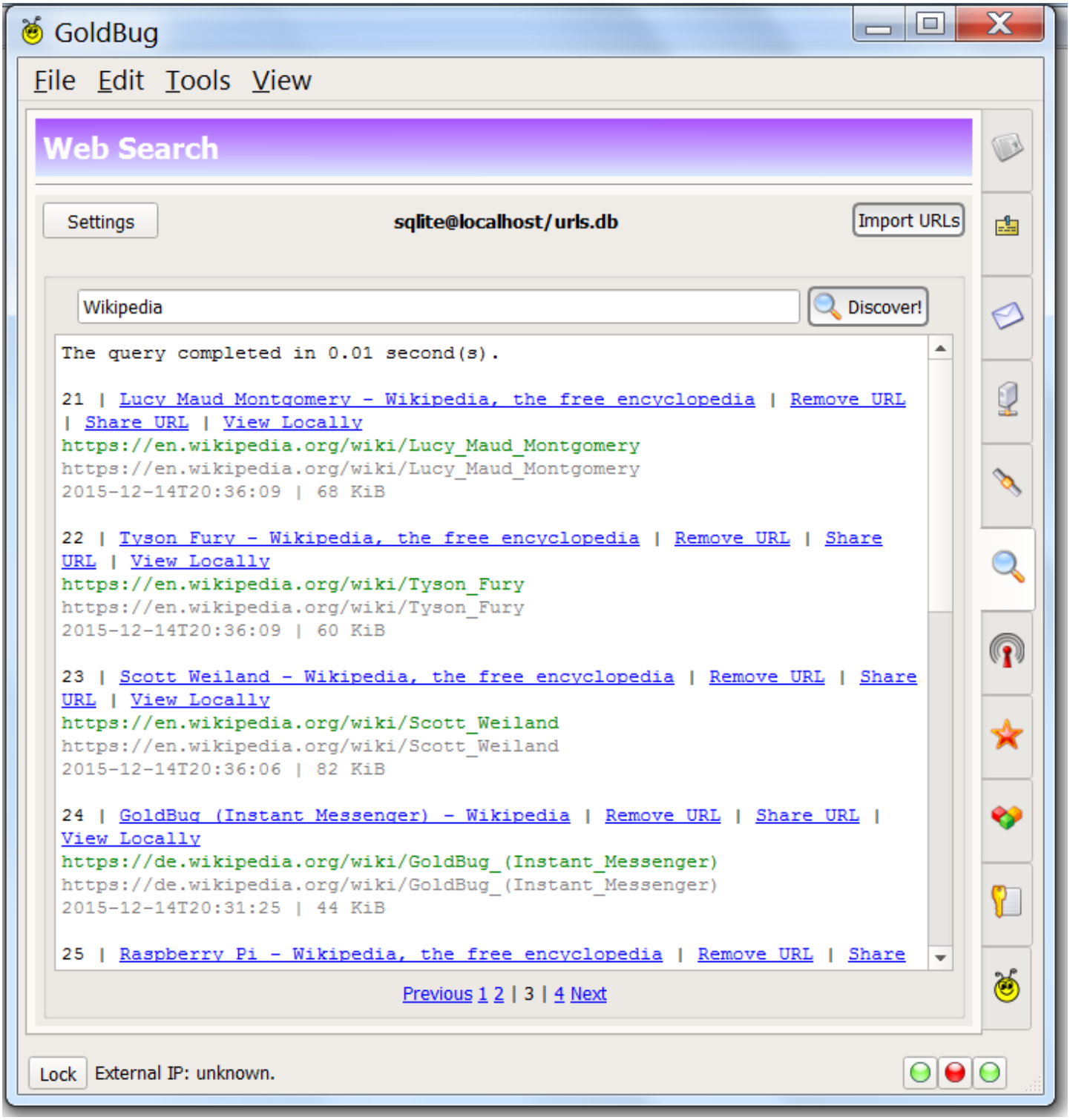
GoldBug is the only (and so far) one of the few handiest p2p distributed search engines like YaCy, Faroo.com, Arado.sf.net or Grub (which was once known by Wikia-Search), which is able to handle the transfer of the URLs over encrypted connections into a distributed F2F or P2P network.

The claim of the web search function in GoldBug is not only to offer an open-source programming of the search engine or the sorting algorithm, but also to handle the repository of URLs open source, so that each participant can download the entire URL-Database. Third, finally, transfers and database storage take place in an encrypted environment. An innovative and exemplary model to search in encrypted databases.

Website titles, keywords and the URL itself are stored encrypted in a SQLite or PostgreSQL database and linked together via the Echo protocol.

A user can use a crawler or RSS feed to store his web pages and URLs in a searchable database repository and share them with other nodes.

Figure: Web search with GoldBug in the URL database



The user can design his own search engine: for example, with 15 GB of URLs in the database on his machine, the user can certainly achieve interesting search results for new websites that his friends find interesting and entered into the p2p network.

But also as a local database for own bookmarks or an own crawl of a dedicated domain, the URL database can be used.

The web search in the URL repository remains anonymous, because the GoldBug URL search generates in other nodes no announcement of the search words, so-called “query hits”.

GoldBug converts the search words into a hash and searches the local databases to see if they contain this hash. Then there is also the hash of the URLs that contain this keyword. The URL

database is then searched for the hash of the URL.

The databases are also encrypted, so that after the search process also a decryption process is connected. Finally, the search results are generated and shown to the user. The UI currently sorts the results for one or more search words for simplicity, such that the most recent URLs are displayed first at the top.

If you want to create an open-source search algorithm for sorting URL results, GoldBug will provide the open source code base for this function in order not only to develop an own algorithm model, but also to subject it to a practical test.

The distribution of website URLs does not happen via central servers, but is organized via the encrypted echo protocol decentralized between the participants: Two or more users exchange their URL keys and then take on the p2p exchange to website URLs, such as own bookmarks, with all their friends. The online exchanged URLs are first collected in main memory and then written to the local database every 10 seconds.

There is also the option of manually importing new URLs into your own local database. This requires the web browser Dooble.sf.net . The first icon in the URL line of the browser allows storing a single URL in an intermediate database: Shared.db. This is then imported by GoldBug with just one click. The Shared.db must be in the installation path of GoldBug and both programs, GoldBug and Dooble, must define in the settings the path of this file.

In order to import a URL of the web page that a user is currently reading from the Web Browser Dooble into GoldBug's own URL database, simply click on the first icon in the URL line of the browser to start the URL to be stored in the URL-DB: Shared.db. Then, in GoldBug, click on the tab "Import" in the tab of the web search.

However, the newer version of the browser Dooble no longer supports this import function of a single URL in GoldBug. Because the new version of the browser Dooble represents a complete reprogramming. which became necessary due to the change in Qt regarding the Webkit module.

In the still available source code of the old Dooble Browser, however, this option can be reactivated with an own compilation. This option should only be mentioned here for a short time, since other developers may also want to import an URL from a (any) browser into an encrypted bookmark database and look at this model.

The idea of making bookmarks shared with friends searchable and locally storable for own history thus remains current.

More efficient, however, are the other methods to import numerous URLs using a crawler or the RSS feed in GoldBug.

But first let's look how to setup the URL database in GoldBug.

12.1 Database Setup

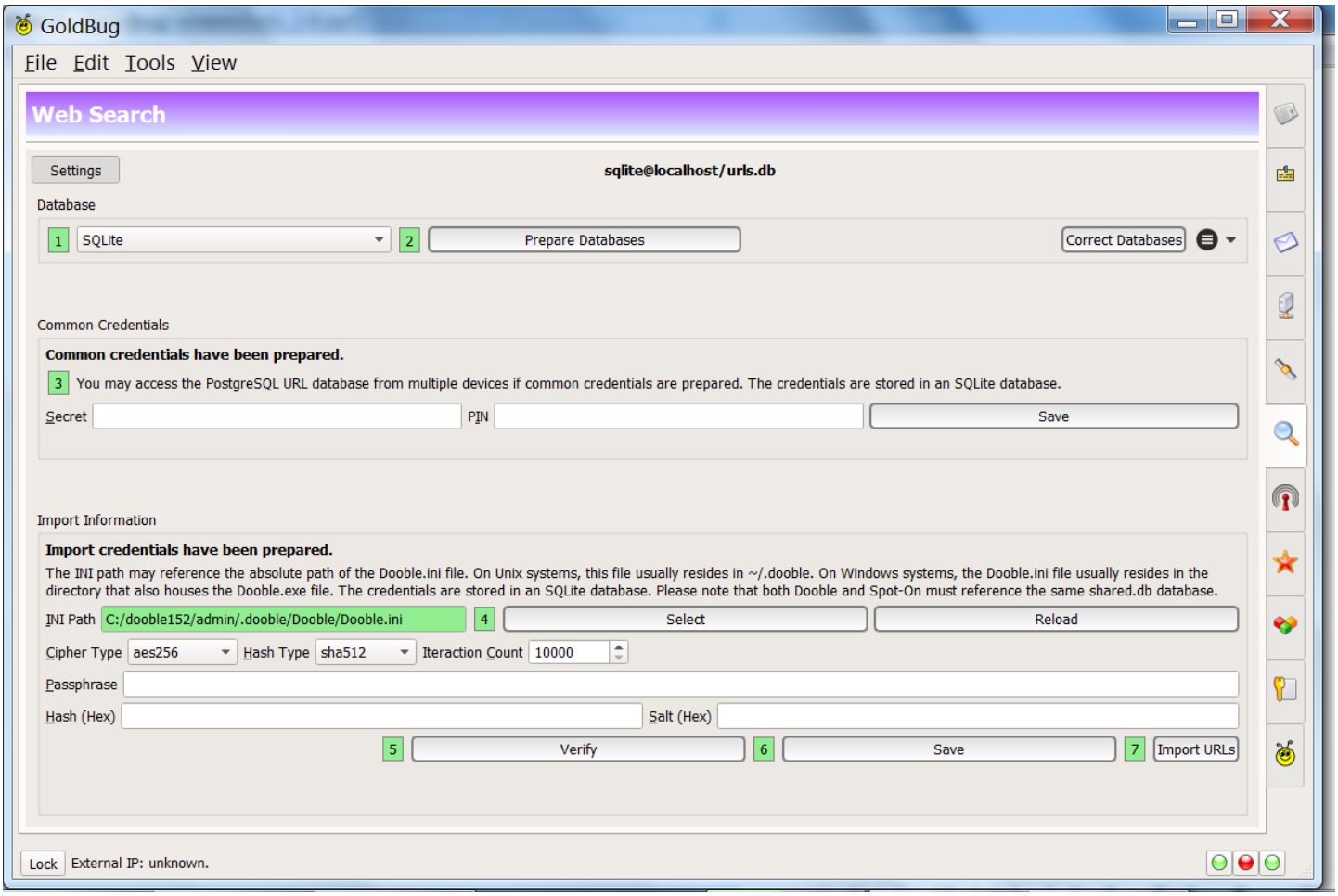
The URLs can optionally be stored in a SQLite or PostgreSQL database. SQLite is the automatically configured database that is also recommended for users with less experience in setting up databases. More advanced users can also contact a PostgreSQL database facility. This has advantages in the network access, the administration of user rights and the handling of large URL data stocks. GoldBug is therefore suitable for creating your own web search, even for teaching purposes, in case those learners are interested in setting up databases.

The URLs are stored in 26x26 or 36x36 databases (2 (16 ^ 2) = 512 tables), which are encrypted. This means that the search takes place in an encrypted database (URLs.db). Searching in encrypted databases is a field of research that has so far received little attention.

12.1.1 SQLite

SQLite is a program library that contains a relational database system. The entire database is in a single file. A client-server architecture is therefore not available.

Figure: Installing the URL database for the URL/Web search



The SQLite library can be directly integrated into appropriate applications so that no additional server software is required. This is also the ultimate difference from other database systems.

Integrating the library extends the application with database functionality without relying on external software packages.

SQLite has some special features over other databases: The library is only a few hundred kilobytes in size. A SQLite database consists of a single file that contains all tables, indexes, views, triggers, and so on. This simplifies the exchange between different systems.

12.1.2. PostgreSQL

PostgreSQL - also known as Postgres - is a free, object-relational database management system (ORDBMS). Its development originated in the 1980s from a database development of the University of California at Berkeley, since 1997, the software is developed by an open-source community.

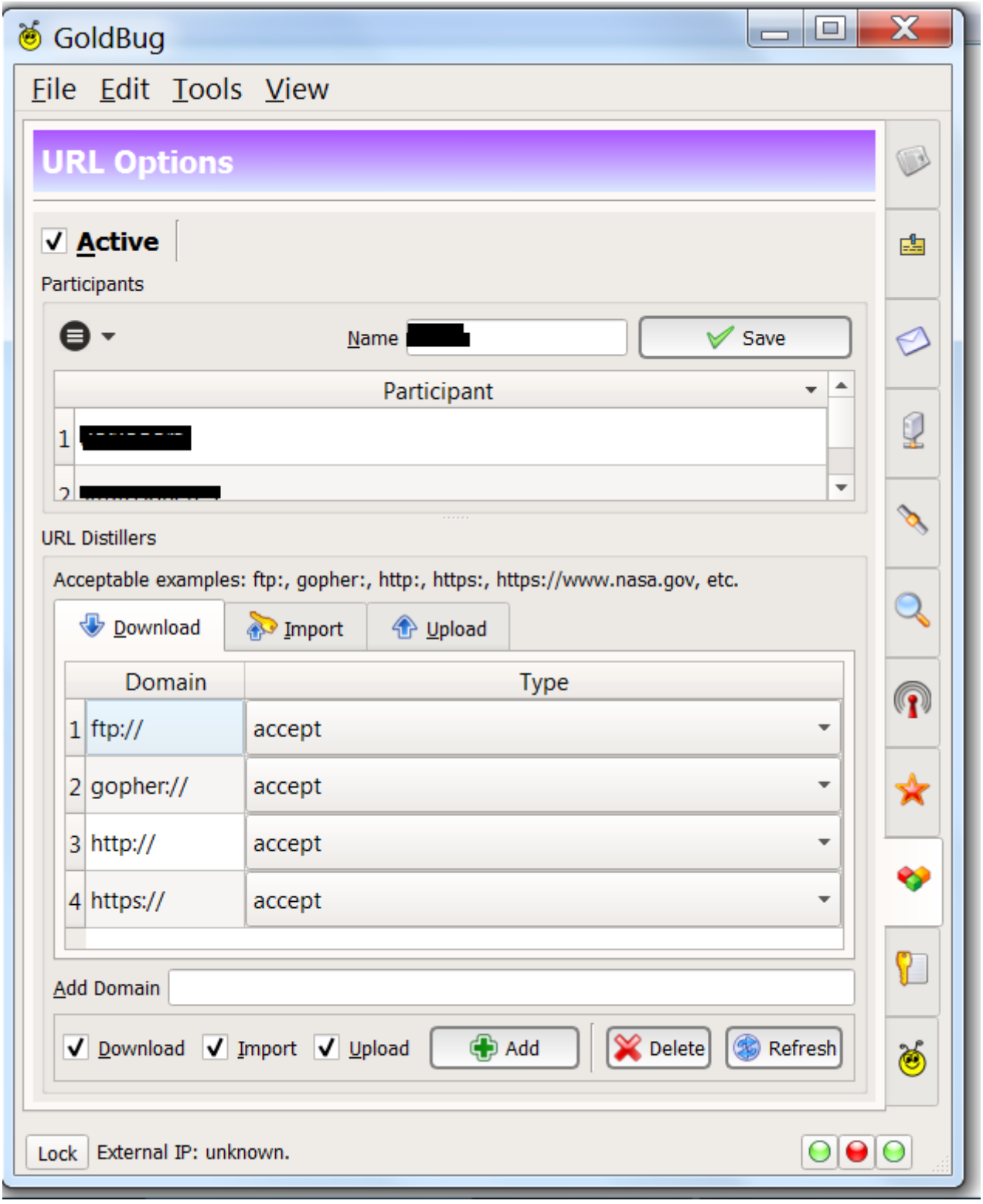
PostgreSQL is largely compliant with the ANSI SQL 2008 SQL standard. PostgreSQL is fully ACID compliant, and supports extensible data types, operators, functions, and aggregates.

Most Linux distributions contain PostgreSQL - Windows and Mac OS X are also supported. Since the setup process of the PostgreSQL database is more extensive, it should also be referred to the manuals of this database with regard to its own p2p capability outside the p2p echo network.

12.2 URL-Filter

If the user now participates in the p2p process of URL exchange, he gets all the URLs that others have added to the system. In order to exclude malicious URLs, the user can also delete URLs in the web search with a single click - or else he uses the URL filter from the beginning, which can be found in its own tab.

Figure: URL Options: Import and Export Filters: URL Distiller



URL filters - so-called distillers - can filter incoming, outgoing and imported data with a blacklist or whitelist. For example, the user can define that only URLs from the domain www.wikipedia.org are allowed or that uploads to URLs to friends only take place from the domain of his university. Also, the user can specify that he does not want to receive URLs of a particular country domain.

In case the user does not want to receive URLs, he just sets the distiller filter to “http: //” with the value “Deny” for the downloads, then these URLs will not be accepted.

Very important: in order for the filter to be active, the filter should be set to “Active” at the top of the check box.

12.3 URL-Community

So that the user can happily exchange URLs and his database grows for web search, he can either manually paste the URL key into the tab “URL Filter” in the participant table; Or, the second option is to send its URL key to a community.

If the user’s friend is also online, and the user uses the “EPKS” tool - Echo Public Key Share - to send his URL key to the “Spot-On URL Community” defined there, his friend receives the URL key of the User automatically transferred online.

This transfer is encrypted using the echo protocol and uses the name of the URL community as symmetric encryption. It is similar to a group chat room (e’IRC/Buzz function) where the URL keys are then sent out and automatically integrated. How EPKS works is described in more detail below.

12.4 Pandamonium Webcrawler

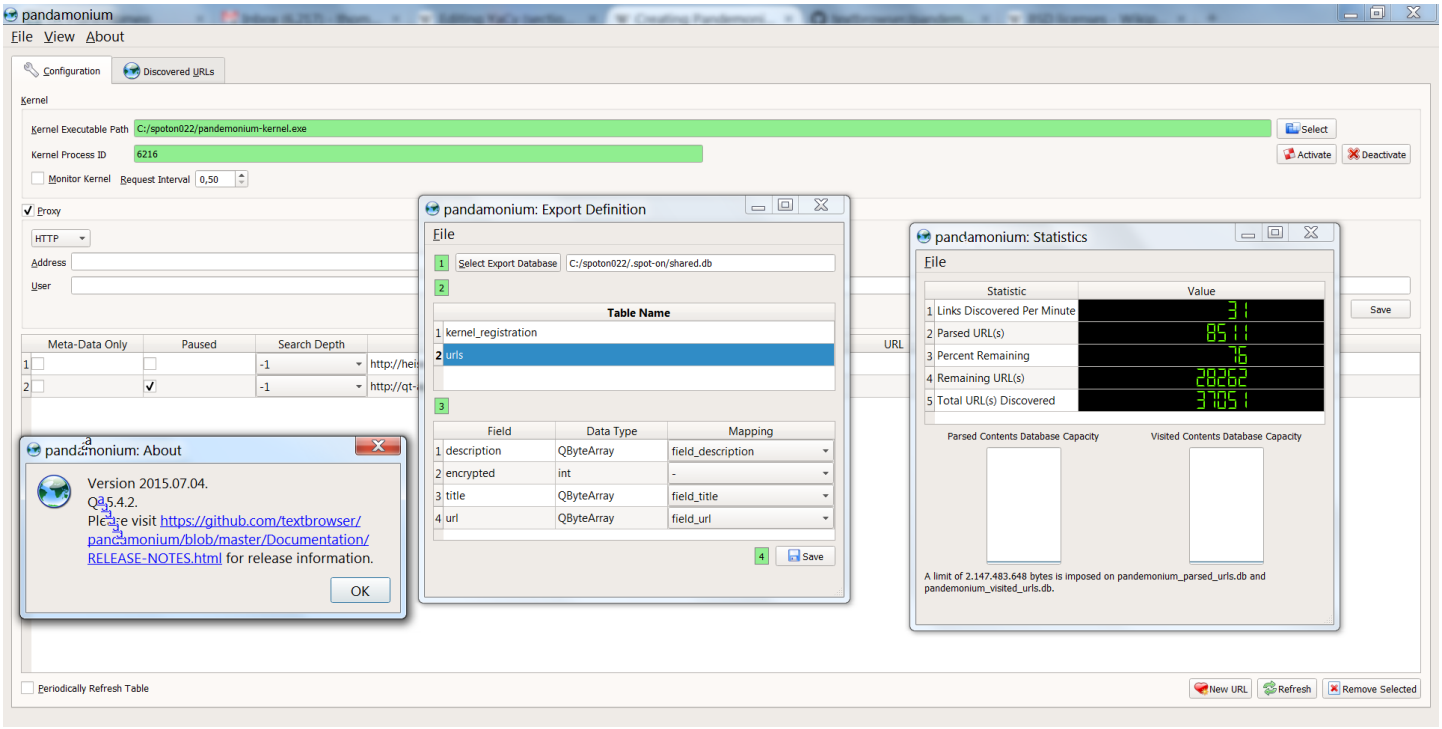
Another import option for URLs is to use the Cawler “Pandamonium”.

The release of GoldBug Version 2.8 (Christmas-Release 2015) is named “Pandamonium Webcrawler Release” and refers to the web crawler named Pandamonium, which has been added as a tool to the URL database feature.

The web crawler scans a domain for all linked URLs and can then index new URLs on the discovered websites and add them to the crawl or index. Pandamonium works (as well as the import from the Dooble Web Browser) via an intermediate Shared.db. The web crawler Pandamonium is also open source and can be downloaded from this URL:

https://github.com/textbrowser/pandamonium

Figure: Pandamonium Web Crawler



The URLs added in this way are then also shared with the friends via encrypted connections or stored encrypted in their own local database as well.

For example, the Pandamonium crawler offers the possibility of importing large amounts of web pages of desired domains for a web search in the client GoldBug.

In addition to the URL, Pandamonium also stores the website as rich text (that means without images) in the database and this database can also be shared with friends. Web browsing in GoldBug allows you to browse web pages locally without having to contact the Internet or the domain to reveal own IP information.

It is almost a new kind and advanced idea of the anonymization network Tor: No longer the website is contacted live via a p2p proxy network, but the URL is searched in a p2p web search or database and the same website can be loaded as rich text, browsed and read locally, such as from a browser cache or proxy.

Java scripts, images and referral URLs as well as IP information are not included. The user is thus protected from the disclosure of his data and can still read the desired web page of an URL if it is present in the shared data. While web pages can also call additional links or leave traces on the anonymization tool Tor due to Javascript, it is preferable for the web crawler Pandamonium to avoid such security risks.

Various revisions of web pages at different call times of the website (Memento) are also supported - both in the crawler as well as in the web search in the GoldBug client. The page viewer of the web search in GoldBug displays various revisions of the web page, if they exist.

The setup of the SQLite database for importing the URLs from the Pandamonium Webcrawler via the shared.db was done in a few steps:

- The user creates a SQLite database in the GoldBug program under Web Search / Settings.
- The user now enters a password for “Common Credentials”. This is a password feature in case third, further applications provide URLs for import.
- Then the user verifies all inputs and starts the import from shared.db, into which he has previously stored the URLs collected by the Pandamonium Webcraler: The import process retrieves the URLs from this file and adds them to the URL database in the GoldBug client (URLs.db).
- Any imported URLs may be shared with the user’s friends online p2p. To do this, the friend’s URL Key should be entered by the user in the Add Friend tab, or he should use the URL sharing community as described above to swap the URL key.

12.5 RSS reader and URL import

The RSS function extends the GoldBug client to an RSS reader. RSS 2.0 feeds are supported. The news URLs are displayed in a timeline so that the most recent message is always on top.

In addition, the news URLs are indexed, i.e. prepared for local web search in GoldBug. The import of the encrypted RSS database into the encrypted URL database can be done automatically periodically, or even via a manual import button only on action of the user.

The RSS feature not only makes it easy to read selected news portals on a news page, but also manually or automatically import the new URLs into an own local URL database.

Figure: RSS feed reader for importing URLs into the URL database/web search

The indexing of the website uses the 50 longest words of the message (or even more after the user’s setting) to prepare them for the search index of the URL database during import.

For the timeline, the titles of the messages are provided with a hyperlink only when indexing has taken place. The status line shows statistics on how many RSS feeds are subscribed, how many URLs are already indexed, how many URLs from the RSS database were imported into the web search URL database - as well as the total readable messages or URLs in the RSS window.

The messages are read in a Page Viewer, which does not display the messages in a browser, but for reasons of safety only in text form. Java scripts, images and advertising are removed from the pages, it will be displayed only the ASCII characters of the website and the hyperlinks to other websites. With the context menu, URLs and hyperlinks can be manually copied out for a view in the (external) browser.

The RSS reader is proxy-capable and can therefore also preserve the content of the websites behind restrictive environments and then make them available for storage and searching in GoldBug.

A feature that today certainly is also offered by some browsers: to call out or offer the URL history and web pages searchable from the cache of the user.

GoldBug allows this in an encrypted and p2p environment for local storage in a dedicated URL repository.

13 Set up chat / email server

Setting up a chat server or spot-on kernel means setting up a so-called “listener”, according to this technical term.

If the user is in the minimal view of the user interface, a chat & email server or listener is set up as quickly as the tab further above shown establishes an IP connection to a server or neighbor.

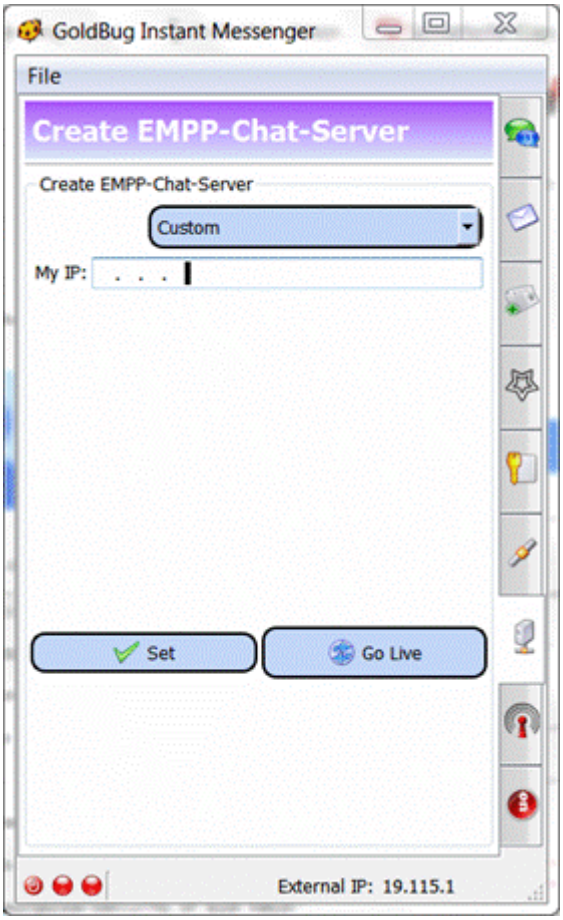
The user does not need advanced server administration skills to run a GoldBug node on their web server, set up a chat server, or even set up an e-mail inbox for themselves and their friends.

In GoldBug only a so-called listener at a defined port must be defined. And that’s possible with just a few clicks. Probably the simplest chat server administration ever compared to other server setups.

13.1 Set up the chat / email server via a listener

As a reminder, on the Connect tab, the user connects their GoldBug to another node or neighbor, and with the Chat Server tab, the user creates a server or listener so that others can connect to it. No matter which method, messages can always be sent if the second or third LED in the status line is green and a neighbor is connected: Either to the other user as a server/listener or the user as a client to the neighbor who offers a listener.

Figure: Creating a Chat Server (Minimal View)



The right (third) LED in the status bar thus indicates that the user has set up his own chat server on his computer.

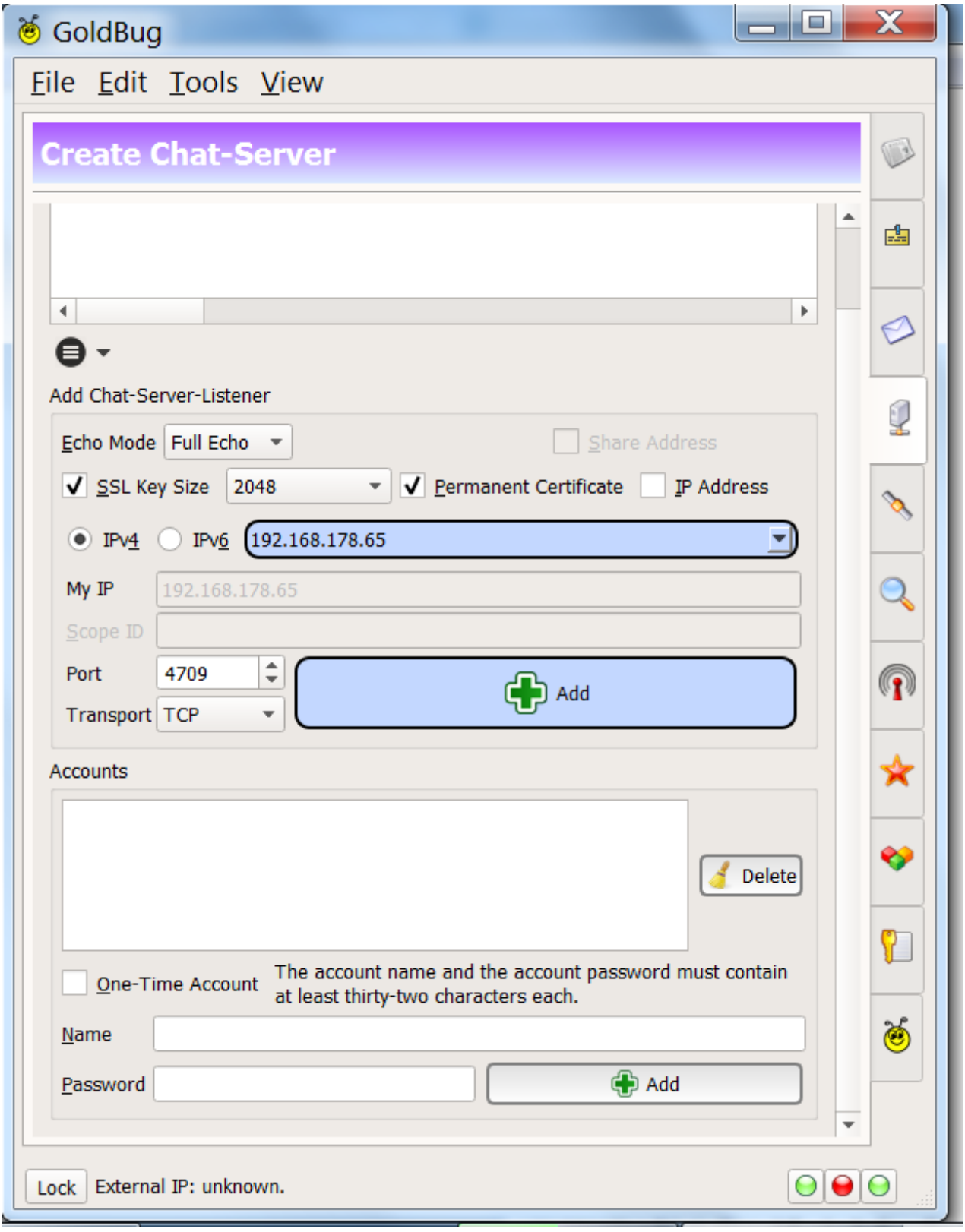
To do this, the user must enter the local IP address of his machine in the “Chat Server” tab. This is not the (external) IP address of the router, but the local network IP address of the device where GoldBug is installed. Again, you get over the pull-down menu a selection, the IP is displayed directly and you can then select the local IP. The port 4710 is then automatically defined again.

Then press the button “Set” and the entry of the listener was successful when the third LED of the status bar is green.

13.1.1. Server broadcast

If the user has a client connected to his listener, or the user in the “connect-neighbor” tab is connected to another chat server or friend on his own, then the user can also right-click the command in the table “Publish information”. Thus, his chat server is communicated over the existing connections to his friends and neighbors as well as their friends. “Publish server” means “Broadcast IP + Port” of the own chat server to its (connected) friends and neighbors. Then the friends can also automatically connect to this chat server. In this case, the user no longer has to communicate an IP address or let his friends enter their own IP address manually. Everything runs automatically and the user’s server is available as a peer to his friends and their friends. It’s that easy to create a chat server and communicate to others on the network.

Figure: Setting up a chat server



The listener or chat server is set up by default for the TCP protocol, but GoldBug is also equipped to set up a listener via the UDP or, thirdly, the SCTP protocol. Both of these latter protocols are ideal for VOIP or streams. Fourth, a chat server/listener via Bluetooth is possible, see the section below.

Therefore, the connection options can also be used to define whether the user's client should connect to the neighbor or another server via TCP, UDP, SCTP or Bluetooth.

The neighbor or listener of the server can do without SSL connections, then the transmission is regulated not over HTTPS, but only over HTTP. This means that an encrypted layer is not required, the encrypted echo capsule is not sent through the HTTPS tunnel, but via HTTP - and still remains encrypted because the echo capsule itself is already encrypted.

13.1.2. Security options

If the user looks at the tab in the maximum view of the user interface, there are further setting options:

For example, a listener may set the security option to generate a permanent SSL certificate. Thus, the Diffie-Hellman key exchange or negotiation process existing in SSL is not renegotiated in every session, but an attacker would already have to know a negotiation process in the past in order to intervene here.

However, it may be that the server or listener renews its SSL certificate, so it may make sense to allow exceptions if you want to make a connection easier and do not want to perfect that extra level of security.

Likewise, one can define the key size for the SSL connection and also determine that connections below a certain SSL key size are not established at all. One time it is defined, what the neighbor should offer in regard of the SSL key size, and the other time is defined, which key size the user expects from a server or neighbor.

Finally, there is the option that the client determines if he connects to the neighbor with full or half echo. At half echo, the message packet is only sent to the neighbor one hop over the direct connection. Assuming the user’s friend has the web server set up and sitting in front of it and the user does not want his echo packets to go to third parties and his friends, then the user can define with the half echo that his packets received by the server are not be distributed further. The two users chat via a direct IP connection. Both participants see the IP address of the friend and the chat partner at Half Echo. In the Full Echo, the chat friend does not have to be an administrator of the node, but can connect multiple clients like a central chat server.

Security options allow you to define the SSL key size when creating a chat server/listener in Advanced View, as well as maintaining a permanent SLL certificate.

Also, the user - if he has a permanent, stable IP address – can integrate this into the SSL certificate.

These three measures make it harder for attackers to exchange or “fake” the SSL certificate - because it would be immediately recognized if a different certificate was to be used as the original one: for example, the client would not be a new, but the old, permanent certificate expected or because the IP address is missing or inconsistent. The SSL key size also defines this.

13.1.3. Proxy and firewall annotations

If the user wants to run GoldBug as a client via a proxy in the company, behind a firewall or a proxy of the university or via the anonymization network Tor, he can insert the proxy details for his neighbor.

As a client, the user can connect to any IT environment thanks to the HTTP protocol, even if he can surf in that environment with a browser.

That's the advantage of the GoldBug program, which means that wherever users can surf with their browsers, they can also email and chat with GoldBug Messenger because of the HTTPS or POPTASTIC protocol they use. Many other programs cannot do this, depending on the firewall settings – e.g. from the workplace or in the student residence.

If the user wants to use or test out a proxy e.g. in his company or university with the GoldBug messenger, then this is uncritical, because an SSL/TLS or HTTPS connection is established - which is hardly different for the proxy administrators like an SSL/HTTPS connection to an HTTPS website when doing banking or logging into a web e-mail provider.

It is essential to address a node in the web with the own GoldBug, which may not be limited by the port through a firewall or the proxy. If so, the user may ask their friend to set up the GoldBug chat server on port 80 or port 443 instead of 4710 and provide it with login information for an echo account, if available and deliverable.

Encrypted traffic remains encrypted traffic, and any GoldBug friend or chat server can be reached on the web through ports 443 or 80 or ports, which are regularly opened in the firewall.

Since the echo protocol only requires a simple HTTP connection to a neighbor (and not necessarily a Stun server or a DHT etc.), and thus ideally can be mapped through a proxy, through a firewall or over the Tor network, it's a very simple architecture to operate chat securely through a proxy or a proxy network.

If the user wants to define additional features in the non-minimal view, a further often-used function is that of the echo account.

To do this, the user in the table marks the listener that he has created and then enters the account credentials, i.e. the name and password. The user then tells his friend what the account name and password are, and when the friend establishes the neighbor contact, the friend will be asked via a pop-up window to enter these credentials.

Similarly, the user can also choose between IPV4 and IPV6 again, if he wants to create a listener/chat server. Also, multiple chat servers can be created by choosing a different port. The user can create different listeners with port 4710 or 80 or 443 and decide whether he wants to define these listeners for friends with an echo account (Friend mode), or for easier to build connections that runs in peer mode without account login.

Echo accounts thus define whether the user builds an F2F network or a P2P network, because with the account credentials the user creates a web-of-trust with which only his trusted friends

can connect with the login password.

13.1.4. GoldBug as Lan Messenger

If the user operates a peer, for example, at a LAN party of a closed network with the IP broadcast function, he can inform all participants that his node has opened a listener for the guests. Thanks to the UDP protocol, however, the GoldBug Messenger also works directly like a Lan messenger within a closed user group of the LAN.

For this, the LAN listener is already defined as a neighbor in the neighbor table (IP: 239.255.43.21). This is just to activate and other GoldBug installations in the same Windows network are then automatically found for a connection.

13.2 Server / Listener Creation Home behind a router / Nat

If the user does not have his own server on the web or does not find a general neighbor on the web, he can also set up his own chat server at home behind his own router and forward the port in the router. His friend can then connect directly to his listener as a client.

However, one of them has to create a listener if they both sit behind a firewall or do not use a chat server on the web. So if the user wants to create a server behind his router / Nat at home, as mentioned the local IP address of the machine for the listener is to take, e.g. 192.168.121.1. Then the user must also forward the port in his router, i.e. port 4710 must be forwarded by the router to 192.168.121.1: 4710. Furthermore, the kernel - Spot-on-Kernel.exe - as well as the GoldBug.exe should be allowed as an exception in the Windows Firewall. If the user has forwarded everything correctly, the friend can connect to the client's (external) IP address (see, for example, www.whatismyip.com) and port 4710.

It is only important that the router of the user forwards the contact attempt from the Internet at the defined port to his local machine. This is a common and secure procedure and does not open any access to a computer, but over the port and the application is defined as with many other programs that only packages in this sense are allowed.

The user can and must define this by himself and GoldBug does not contain any code that automatically forwards ports in the router, or opens or even automatically sets up a listener!

Thus, in GoldBug it is more secure and requirement regulated than in other applications that configure themselves in the interest of user-friendliness and reduce the effort and provision of background automation. - But do this also for users who know the technical details of port forwarding within the router and for the creation of a listener, and need no automatic definition, opening and forwarding of ports.

13.3 Use of GoldBug in the TOR network

If the user wants to operate his GoldBug chat through the Tor network, this will also be very comfortable, so a Tor Exit Node will only see the encryption text of GoldBug. Here, the chat server is again in the normal web outside the Tor network.

So far, Tor cannot establish HTTPS connections at the exit node of the Tor network, but pass-through of encrypted packets from two GoldBug instances should be possible: GB -> Tor -> Tor -> Tor -> GB. An HTTP listener can also be set up for the TOR network. This is an ideal integration of both applications.

13.4 Spot-On Server

In addition to GoldBug, which is an alternative user interface to the Spot-On kernel, there is also the original Spot-On user interface under spot-on.sf.net. A chat server listener can also be set up with that server software called “Spot-On”.

13.5 Spot-On Lite Server as Deamon

If you want to administer a chat server without a user interface, thus using the chat server as a kernel daemon on a web server, you can view the other server software for echo clients at github.com/textbrowser/spot-on-lite.

13.6 SmokeStack Server on Android

The simplest option at home in the LAN or even with port forwarding in the router to set up a chat server is to install the app SmokeStack on an Android device. This Android server can also forward the echo packages of GoldBug clients. Available at:
github.com/textbrowser/smokestack

13.7 Bluetooth server

Finally, a chat server/listener via Bluetooth is possible (since version 2.8, depending on Qt currently only for Linux). With Bluetooth, it is possible, for example, to connect the devices BT-wirelessly via the echo protocol at a LAN party. This option can be very crucial if there is no Internet or infrastructure left.

Figure: Bluetooth chat server

![Abbildung: Bluetooth Chat Server] (/images/bluetooth_chatserver.png)

13.8 UDP Server

The User Datagram Protocol, UDP for short, is a minimal, connectionless network protocol that belongs to the transport layer of the Internet protocol family.

The development of UDP began when a simpler protocol was required for the transmission of speech than the previous connection-oriented TCP. A protocol was needed that was only addressing, without securing the data transmission, as this would cause delays in voice transmission.

A three-way handshake, such as TCP (the Transmission Control Protocol) for establishing the connection, would create unnecessary overhead in this case.

UDP is therefore a connectionless, non-reliable and unsecured as well as unprotected transmission protocol. That is, there is no guarantee that a packet once sent will also arrive, that packets arrive in the same order in which they were sent, or that a packet arrives only once at the receiver. An application that uses UDP, therefore, must be insensitive to lost and unsorted packages or even provide appropriate corrective measures and, if necessary, safeguards.

For the Echo Protocol an interesting basis, since the packets are indeed rather undirected in the flow of the network and lost by the multiplication lost UDP packets so again through the redundancy.

13.9. SCTP Server

The Stream Control Transmission Protocol (SCTP) is a reliable, connection-oriented network protocol. As a transport protocol, SCTP is at the same level of the TCP / IP reference model as TCP and UDP.

SCTP realizes the concept of an association: Here, a connection is set up in which several message data streams are transported in order-preserving (with each other but potentially non-order-preserving). In addition, individual, for example, urgent, datagrams may be sent separately and out of line, possibly “overhauling” the in-order data streams.

Also, to use this protocol for the transmission of echo packets is very interesting for the research, since the rather undirected echo packets may experience a more secure transmission with this protocol compared to UDP.

This protocol can also be used to set up a chat server for GoldBug.

13.10 Ncat connection

While other applications always require a server that may be difficult to replicate or manage, GoldBug also has the ability to do without dedicated server software. For this purpose Ncat can be used as follows:

In one exercise, two devices are connected to GoldBug through a RaspberryPi running Debian using NCat. It requires a working network, a RaspberryPi and two devices each with GoldBug.

First, ncat will be installed on the Pi:

> sudo aptitude install nmap Then some SSL material is generated:

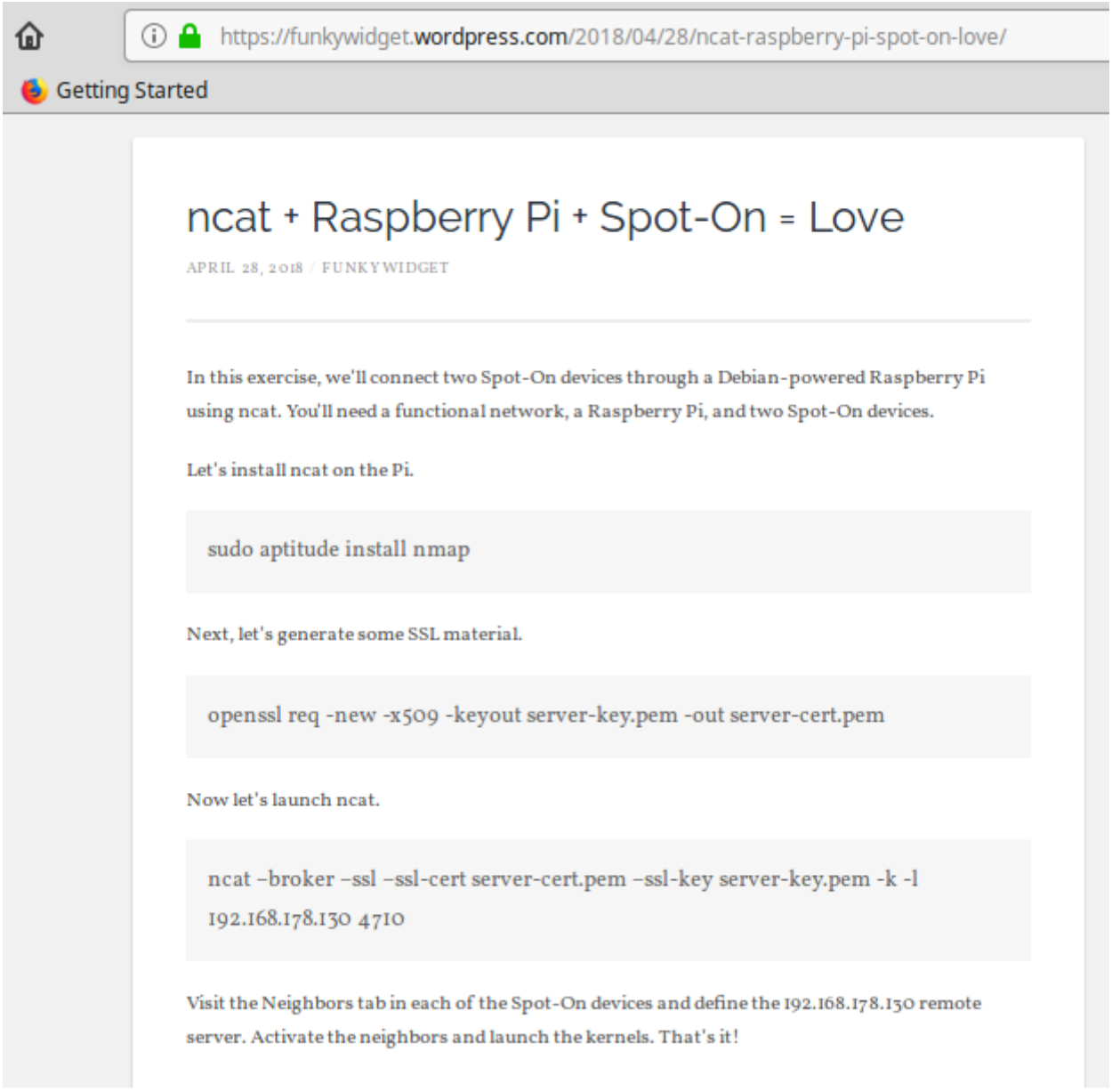
> openssl req -new -x509 -keyout server-key.pem -out server-cert.pem Then ncat is called.

> ncat -broker -ssl -ssl-cert server-cert.pem -ssl-key server-key.pem -k -l 192.168.178.130 4710 Now the user visits the neighbor / server tab in GoldBug and defines the remote server at 192.168.178.130.

If the neighboring devices have been activated and the kernels are turned on, the connection already exists.

A nice exercise to test a network and Ncat for encrypted communication.

Figure: Exercise connection via Ncat



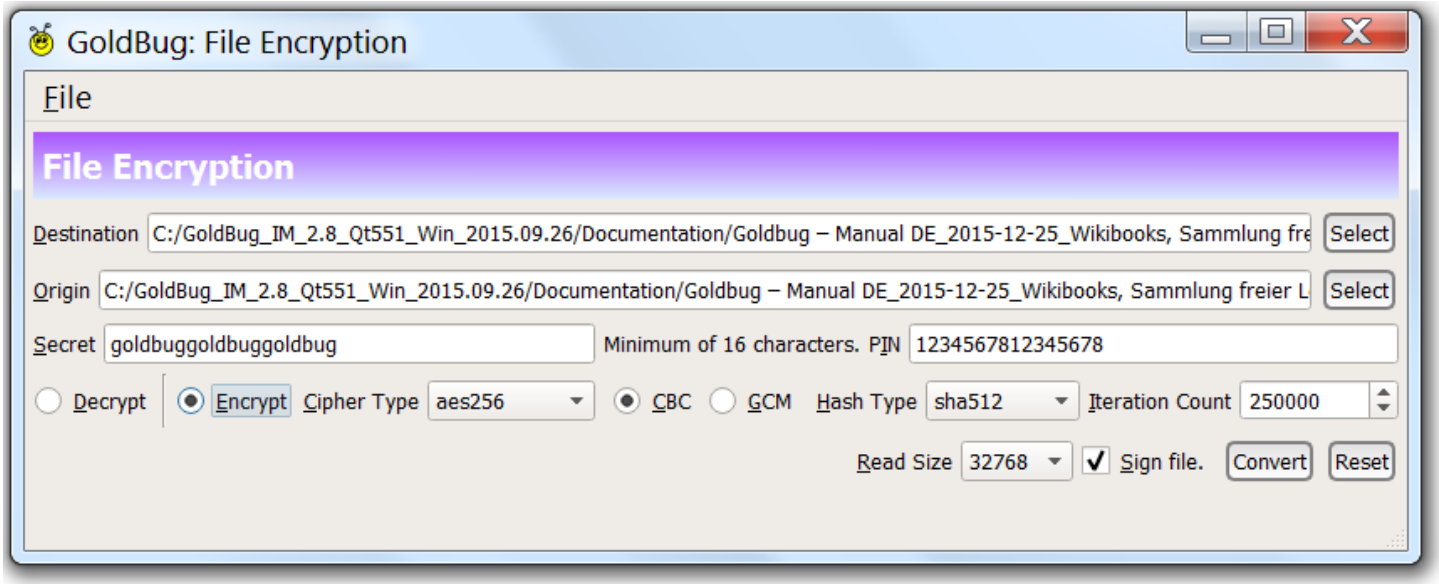
14 Tools

In addition to the regular functions, GoldBug Messenger also has several tools that offer useful features. These include, in particular, the functions of file encryption (File Encryptor), another tool for converting normal text and cipher text (Rosetta-CryptoPad), and the EPKS tool, with which the public keys for encryption are transmitted online can be. Furthermore, the pass-through functionality as well as the tools for statistics and analyzes should be mentioned.

14.1 Tool: Encryption of files

GoldBug has additional encryption tools. In the main menu under tools the user finds the tool for encrypting files on his hard disk (“File Encryption Tool”).

Figure: File Encryptor - file encryption tool



This allows the user to specify a file from the hard drive, then specify the same path and choose any extension or change of file name - then enter password and pin (both of course again at least 16 characters) and define with the radio select buttons, whether the file should be encrypted or decrypted.

Cipher and hash-type are also definable as well as that a signature in the encryption can optionally be installed to ensure that the encryption was made only by the defined user (and nobody else).

The file encryption tool is an offer to replace potentially insecure Truecrypt containers, or to encrypt them or to back up individual files before the user transfers them - whether as an email in GoldBug, via StarBeam in GoldBug or over conventional, insecure ways - or simply to encrypt them on disk or when stored in online cloud stores like Dropbox or Megaupload.

14.2 Tool: The Rosetta CryptoPad

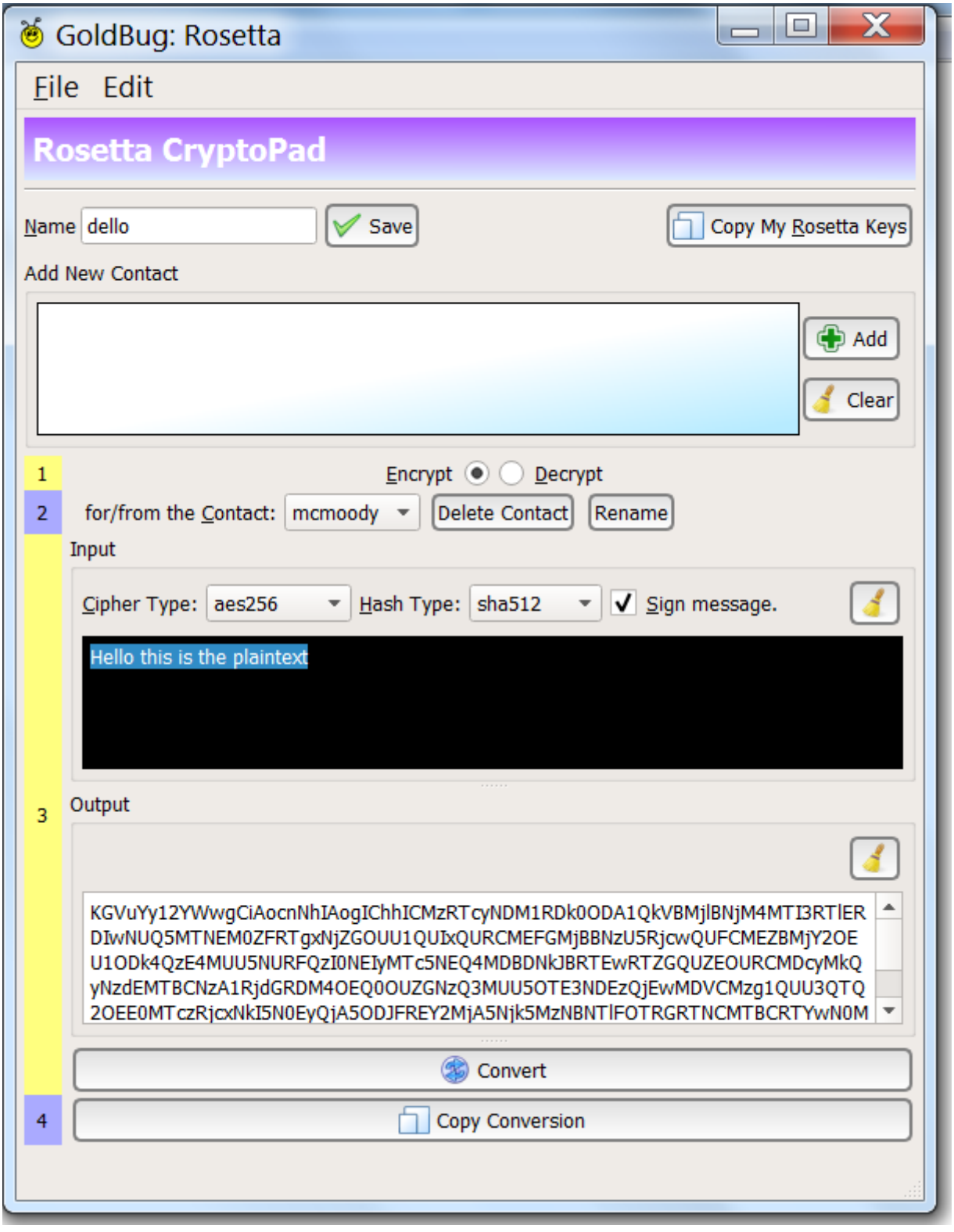
The tool Rosetta CryptoPad takes its name from the stone of Rosette, which stands in the museum in London. It is considered a translation tool for Egyptian hieroglyphs in other

languages.

The Rosetta CryptoPad included in GoldBug has its own key - as well as chat and e-mail, and all other functions have their own keys like this.

The user also exchanges the Rosetta key with a friend, then enters text into the CryptoPad, selects the friend and, whether it is encryption or decryption, - and press the “Converge” button.

Figure: Encryption of text with the Rosetta Crypto Pad



Then the bottom of the output is displayed as ciphertext and this the user can easily copy with the copy function and send via conventional online communication channels such as @-e-mail or another chat. Web boards or paste bins can also be used by the user as a place for encrypted communication.

It is, so to speak, “slow chat” by a manual encryption of the chat text (although the encryption is faster than the copy paste in other instances).

The Rosetta CryptoPad is an alternative to GnuPG (though it is also based on the GnuPG library libgcrypt).

This method of slow chat also shows that applications that rely on encrypting each individual email are an inconvenient method. Who wants to select the recipient for every e-mail and chat message, encrypt the message, decide whether the signature key should still be added or not before the message is sent?

Figure: ROSETTA crypto pad conversion

GoldBug has the general advantage of exchanging the key once with the friend during set up and then everything is encrypted at all times and the entire communication is transferred within

the chosen encryption, with temporary keys and end-to-end passphrases (Geminis der Calling function) can be renewed instantaneously at any time.

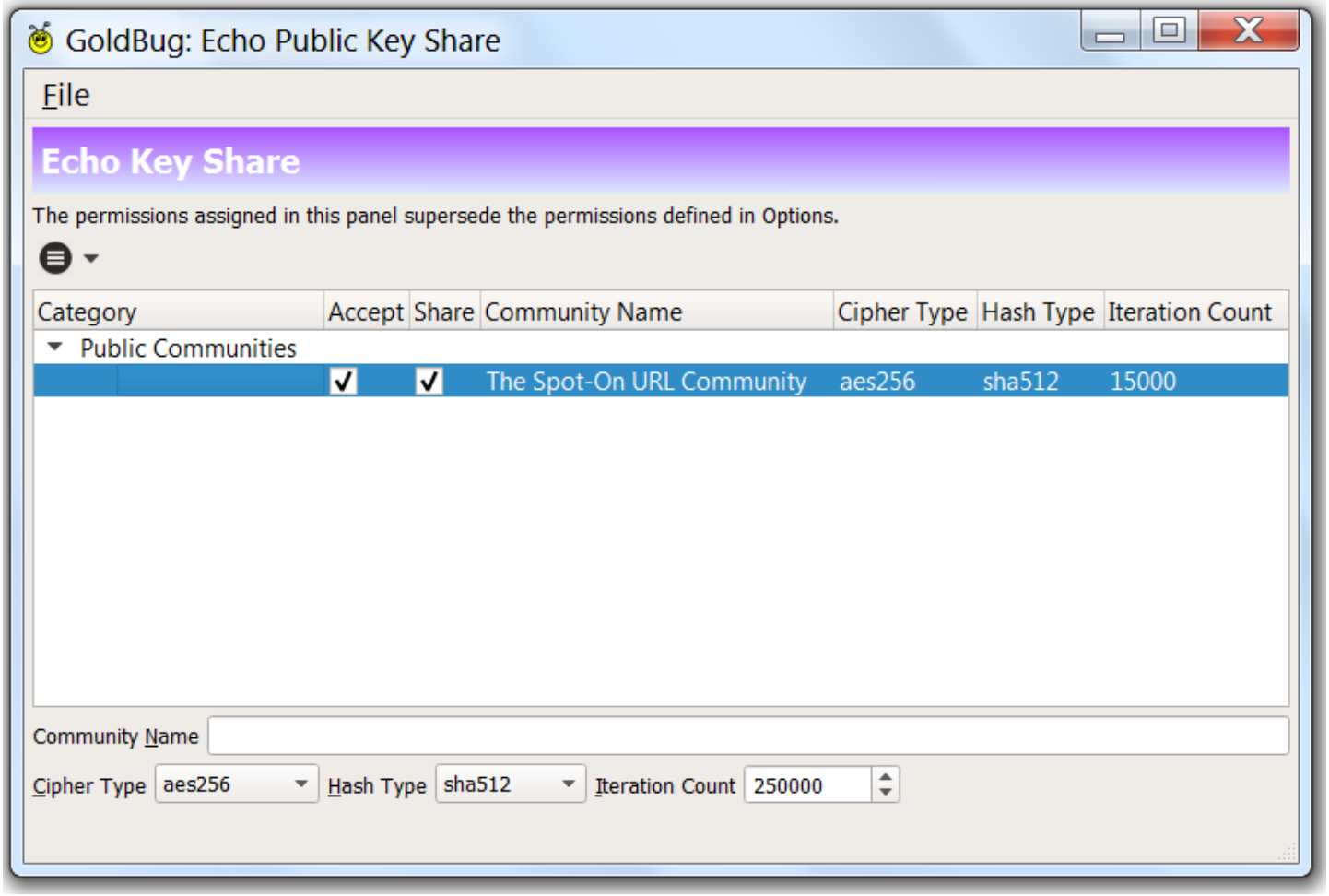
14.3 Tool: Echo Public Key Share (EPKS)

When it comes to encryption, there is always the central question of how to safely transport the key to the friend.

Some architectures use key servers in which the user can set their public keys. This seems logical, after all it is a public key. Nevertheless, the key servers also have massive disadvantages, so you do not know if you have found the right key in it or if this is even up to date.

The Echo Public Key Share function makes it very easy to transfer keys in the GoldBug Messenger.

Figure: EPKS - Echo Public Key Sharing



For this purpose, a symmetric key is defined with a community name in the p2p network of the echo protocol, through which all participants who know the community name can exchange the public keys.

The tool is linked via the main menu and opens a new pop-up window.

An example of a community is already there by default for the exchange of URL keys. The user sends his URL key to this community and all other subscribers who are currently online in the p2p network receive this key.

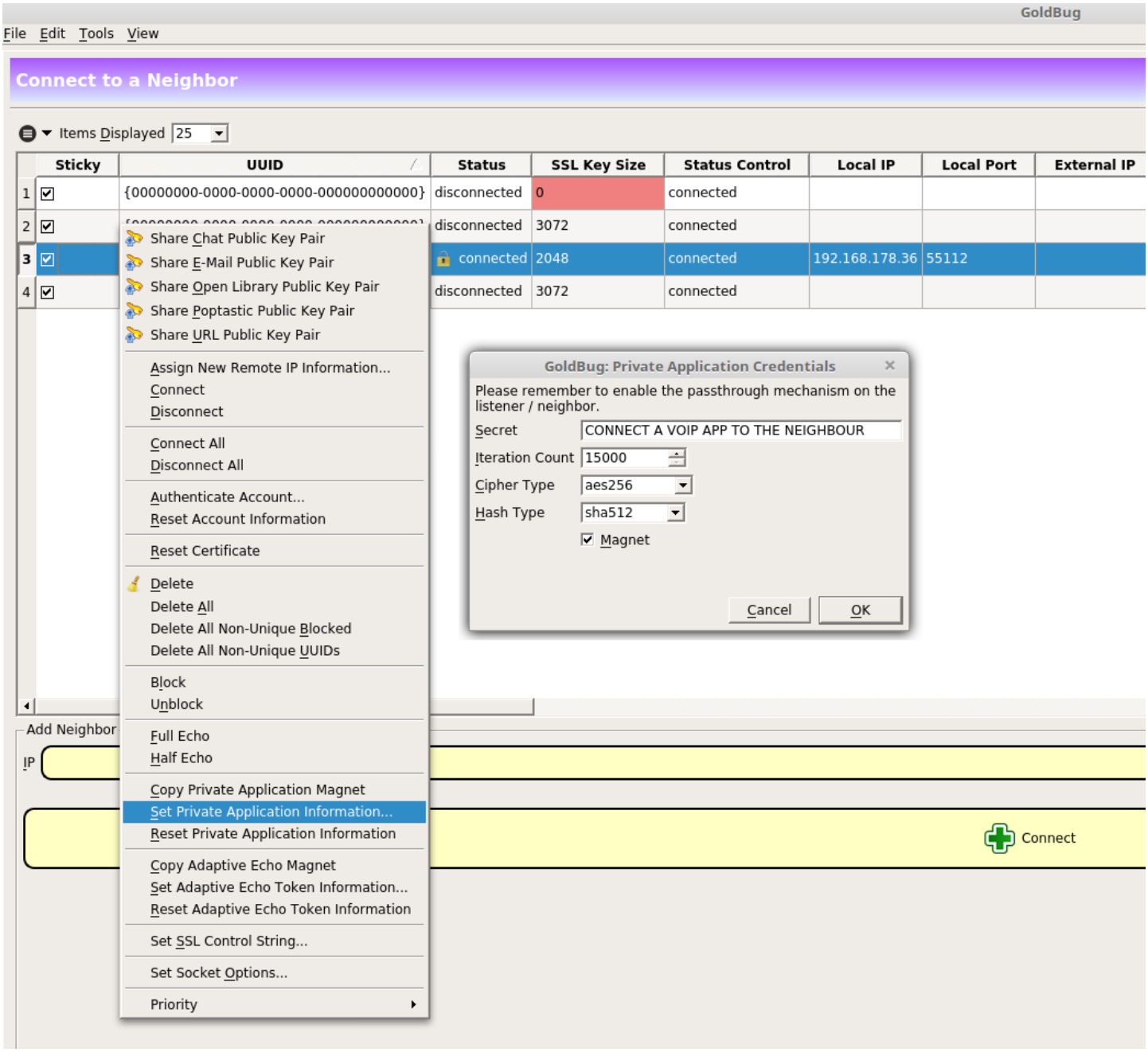
It is a key exchange over a symmetrically encrypted channel, where the password for end-to-end encryption is the name of the community. All users who know the name of the community will be able to receive and add the keys that users put into the channel to their program.

14.4 Pass-Through functionality

If two GoldBug clients have an existing connection over the Internet, this connection can be used as a tunnel to pass the data of another application through this tunnel.

For this, the proxy of GoldBug is addressed with the pass-through functionality.

Figure: GoldBug as proxy: pass-through



This is an interesting feature to protect two clients of another program without encryption over the Internet with the encrypted connection via GoldBug.

Application => GB => GB-Server => GB => Application

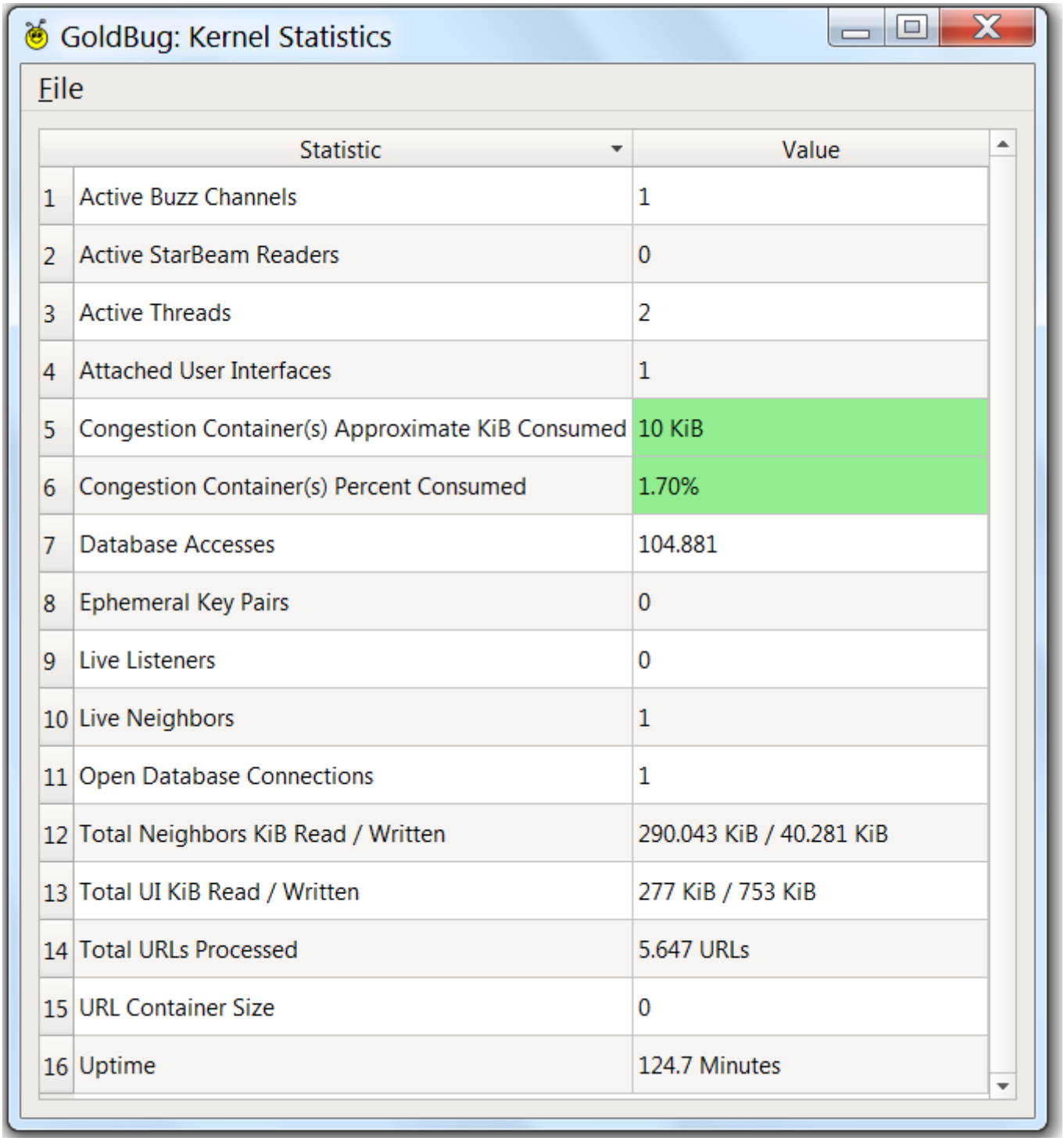
For another application, so to speak, a VPN tunnel is set up, which can even be equipped with McEliece or one of the other encryption algorithms. As long as no VPN provider offers McEliece encryption from start to end, the pass-through functionality, which GoldBug represents as a kind of VPN tool, is the right choice for a test. So far no further tunnel software is known which applies McEliece.

The application to be connected should be tolerant to the order of the sent packets. It is an interesting research test that can be conducted with several possible applications.

14.5 Display analyzes and statistics

In addition to statistics overviews also analysis tools are included in GoldBug, such as the above-mentioned StarBeam Analyzer. The listener and server tables also contain a lot of data information about sent packets, as well as statistics for the URL database.

Figure: Display of statistics



	Statistic	Value
1	Active Buzz Channels	1
2	Active StarBeam Readers	0
3	Active Threads	2
4	Attached User Interfaces	1
5	Congestion Container(s) Approximate KiB Consumed	10 KiB
6	Congestion Container(s) Percent Consumed	1.70%
7	Database Accesses	104.881
8	Ephemeral Key Pairs	0
9	Live Listeners	0
10	Live Neighbors	1
11	Open Database Connections	1
12	Total Neighbors KiB Read / Written	290.043 KiB / 40.281 KiB
13	Total UI KiB Read / Written	277 KiB / 753 KiB
14	Total URLs Processed	5.647 URLs
15	URL Container Size	0
16	Uptime	124.7 Minutes

In addition to the usual user interface, GoldBug can also be installed in console form e.g. on a RasperryPi and retrieve the statistics overview with a corresponding command.

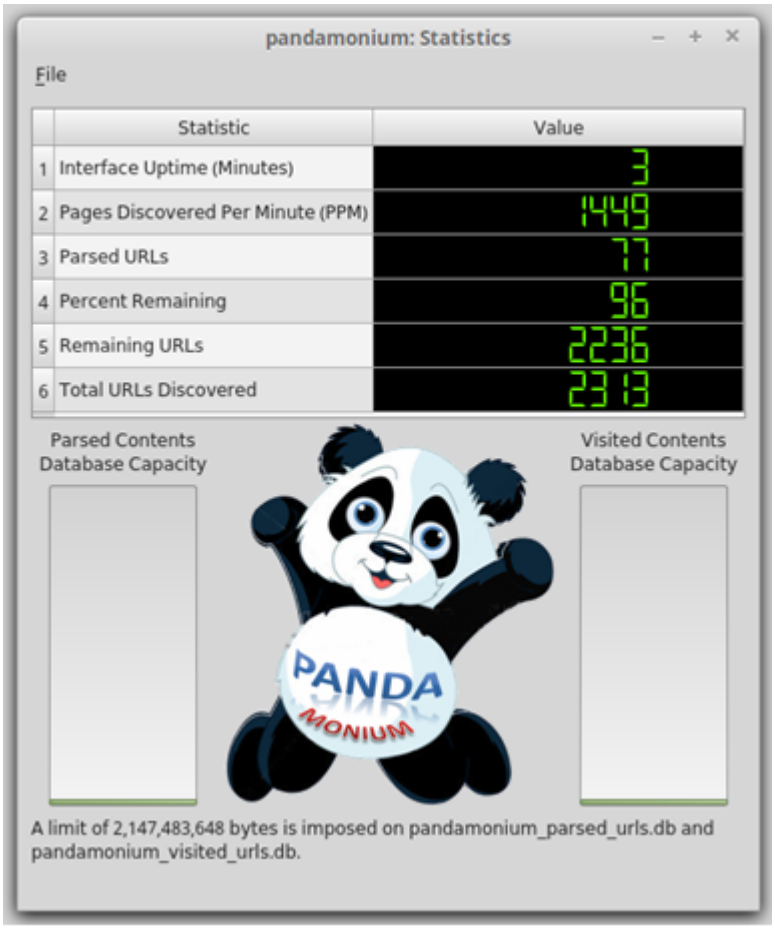
Figure: Statistics console on a Raspberry Pi

GoldBug Messenger on the Raspberry PI - Statistics over Console

```
pi@snoopy:~ $ uname -a
Linux snoopy 4.1.13-v7+ #826 SMP PREEMPT Fri Nov 13 20:19:03 GMT 2015 armv7l GNU/Linux
pi@snoopy:~ $ sqlite3 .spot-on/kernel.db
SQLite version 3.8.7.1 2014-10-29 13:59:56
Enter ".help" for usage hints.
sqlite> .d
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE kernel_gui_server (port INTEGER PRIMARY KEY NOT NULL CHECK (port >= 0 AND port <= 65535));
INSERT INTO "kernel_gui_server" VALUES(38328);
CREATE TABLE kernel_statistics (statistic TEXT PRIMARY KEY NOT NULL, value TEXT);
INSERT INTO "kernel_statistics" VALUES('Active Buzz Channels','0');
INSERT INTO "kernel_statistics" VALUES('Active StarBeam Readers','0');
INSERT INTO "kernel_statistics" VALUES('Active Threads','1');
INSERT INTO "kernel_statistics" VALUES('Attached User Interfaces','0');
INSERT INTO "kernel_statistics" VALUES('Congestion Container(s) Approximate MiB Consumed','2 MiB');
INSERT INTO "kernel_statistics" VALUES('Congestion Container(s) Percent Consumed','0.46%');
INSERT INTO "kernel_statistics" VALUES('Database Accesses','465,717');
INSERT INTO "kernel_statistics" VALUES('Ephemeral Key Pairs','0');
INSERT INTO "kernel_statistics" VALUES('Live Listeners','1');
INSERT INTO "kernel_statistics" VALUES('Live Neighbors','3');
INSERT INTO "kernel_statistics" VALUES('Open Database Connections','1');
INSERT INTO "kernel_statistics" VALUES('Total URLs Processed','0 URLs');
INSERT INTO "kernel_statistics" VALUES('URL Container Size','0');
INSERT INTO "kernel_statistics" VALUES('Uptime','875.2 Minutes');
CREATE TRIGGER kernel_gui_server_trigger BEFORE INSERT ON kernel_gui_server BEGIN DELETE FROM kernel_gui_server; END;
COMMIT;
```

The Pandamonium URL Webcrawler also has corresponding statistics.

Figure: Pandamonium Web Crawler Stats



15 BIG SEVEN STUDY: Crypto-Messenger-Audit

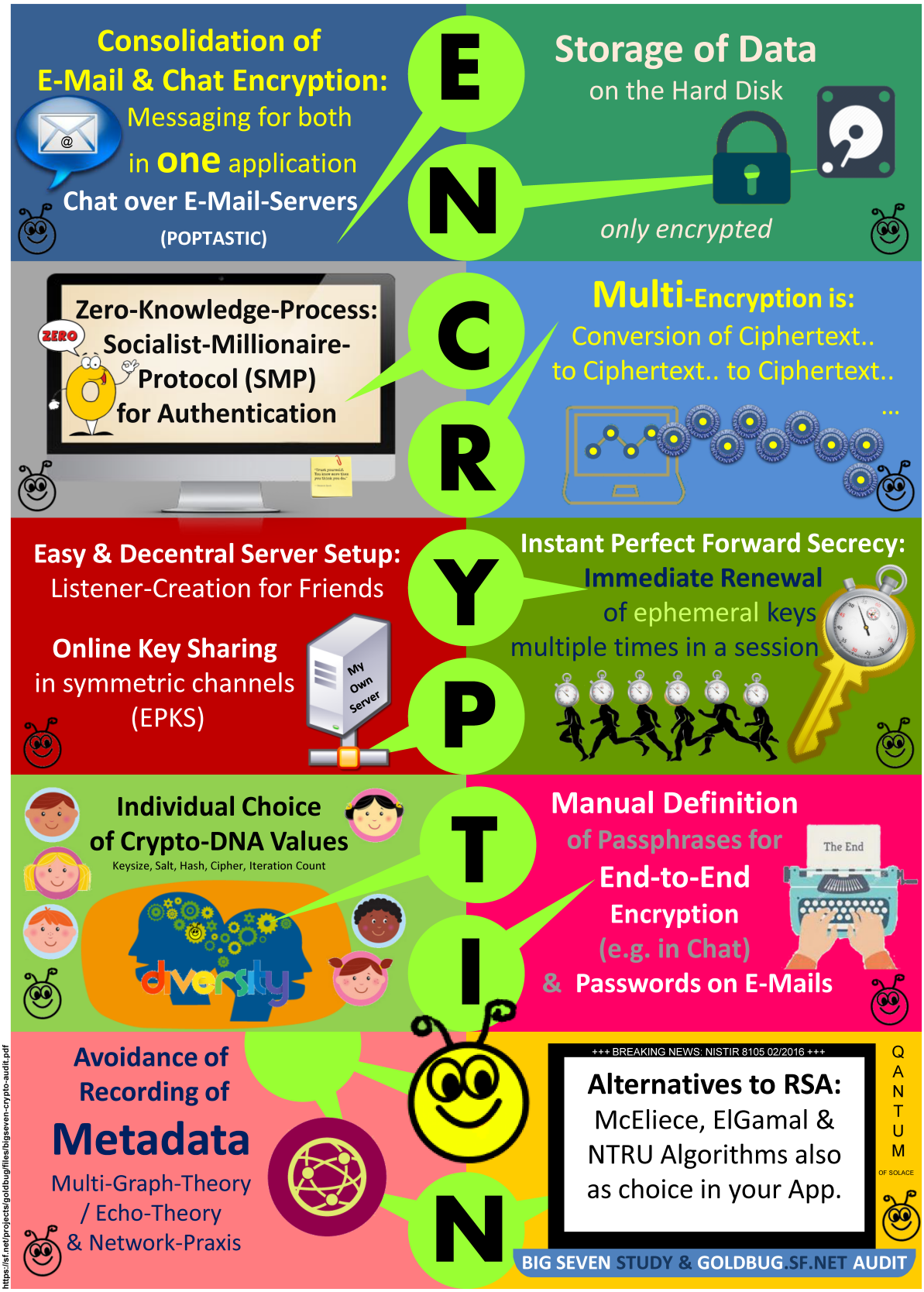
GoldBug was considered by David Adams and Ann-Kathrin Maier as one of the BIG SEVEN crypto messengers among seven open-source messengers, which in an international IT audit-oriented evaluation is regarded more than audit-compliant in more than 20 dimensions and overall trustworthy.

Also the numerous code reviews gave hints in regard of an excellent programming. The “10 Trends in Crypto-Messaging” (see Adams / Maier 2016) identified by all seven messengers core competencies in particular the possibility in GoldBug to set up remote chat servers as well as manual definitions of end-to-end encrypting to make symmetrical passwords.

Figure: Trends in Crypto after the Big Seven Crypto Study (2016)

10 Trends in **Crypto Messaging**

A Study on the open source Applications GoldBug, CryptoCat, OTR+XMPP, RetroShare, Signal, Surespot and Tox.



Adams, D. / Maier, A.K. (2016)

In addition to the auditing of the source code, the architecture and processes of encryption as well as the functions in GoldBug, there are numerous other topics on which future research can

orientate itself. As an example, the following questions should be mentioned for further evaluations and research needs, which may play a role in comparison with other processes and applications:

- Is the Application open source?
- Is it a tiered application: kernel and user interface processes.
- Are there proxy capabilities?
- Is it possible to send E-Mail messages to offline friends?
- Is it possible to send E-Mail with encrypted attachments?
- Are there different Keys for different functions in place like Chat, E-Mail, Cryptopad, Filetransfer etc.?
- Is the key stuck to the IP Address of the user?
- How is mutual access authentication defined?
- Are there alternatives to RSA, like Elgamal or NTRU? Can a NTRU-user chat to a RSA-user? Is McEliece given?
- Are there selectable SSL ciphers?
- Are there selectable hash algorithms?
- Just need connectivity, no key exchange, keys are optional?
- Is trust needed, or can it be added as the user define it?
- What about technical simplicity?
- is it possible to determine, who is reading which message?
- Local databases store all info in encrypted .db's?
- Is the authentication of messages optional?
- Can the user communicate without public keys, using Magnets?
- Support for TCP and UDP and SCTP communications?
- Support of multi-layers of encryption
- Are multiple listeners possible?
- Is a multi-threaded kernel given?
- Are there IRC-like group chat channels?
- What about simple IP-based firewalls?
- Do scramblers send out fake messages?
- Is it possible to store messages in friends?
- Is there the option to use an individually defined and manually inserted end-to-end key for communication?
- Is there the option to renew the end-to-end key each time user wants (not only session based)?
- Encrypted file transfer protocol (StarBeam)? -Using a onetime magnet (OTM) for a crypto channel?
- Having ipv6 support?
- Having Qt 5 and up deployed?
- Sending a message to a friend to his dedicated connection and not to all connections?
- Hiding the key exchange online?
- Use several encryption keys for one file transfer?

- Adding a passphrase on a file transfer?

16 The Digital Encryption of Private Communication in the Context of ...

This user manual is not only intended to technically describe the handling of encryption, its processes or the use of the individual tabs and buttons, but also to illustrate the meaning of the encryption as it stands in the light of various basic laws for the protection of private freedom and communication. The following basic laws should therefore be pointed out, which refer to them in their original texts.

16.1 Principles of the protection of private speech, communication and life: Universal Declaration of Human Rights, 1948 (Art. 12)

No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honour and reputation. Everyone has the right to the protection of the law against such interference or attacks.
http://www.un.org/en/documents/udhr/index.shtml#a12 Universal Declaration of Human Rights Universal Declaration of Human Rights

16.2 International Covenant on Civil and Political Rights, 1966 (Art. 17)

1. No one shall be subjected to arbitrary or unlawful interference with his privacy, family, home or correspondence, nor to unlawful attacks on his honour and reputation.
2. Everyone has the right to the protection of the law against such interference or attacks.
   http://www.ohchr.org/EN/ProfessionalInterest/Pages/CCPR.aspx International Covenant on Civil and Political Rights - International Covenant on Civil and Political Rights

16.3 European Convention on Human Rights, 1950 (Art. 8)

1.Everyone has the right to respect for his private and family life, his home and his correspondence. 2.There shall be no interference by a public authority with the exercise of this right except such as is in accordance with the law and is necessary in a democratic society in the interests of national security, public safety or the economic well-being of the country, for the prevention of disorder or crime, for the protection of health or morals, or for the protection of the rights and freedoms of others. http://conventions.coe.int/treaty/en/Treaties/Html/005.htm European Convention on Human Rights - European Convention on Human Rights

16.4 Charter of Fundamental Rights of the European Union, 2000 (Art. 7, 8)

"'Article 7 - Respect for private and family life'" Everyone has the right to respect for his or her private and family life, home and communications.

"'Article 8. Protection of personal data'" 1.Everyone has the right to the protection of personal data concerning him or her. 2.Such data must be processed fairly for specified purposes and on the basis of the consent of the person concerned or some other legitimate basis laid down by law. Everyone has the right of access to data which has been collected concerning him or her, and the right to have it rectified. 3.Compliance with these rules shall be subject to control by an independent authority. Charter of Fundamental Rights of the European Union - Charter of Fundamental Rights of the European Union (Wikisource) Charter of Fundamental Rights of the European Union - Charter of Fundamental Rights of the European Union

16.5 Basic Law e.g. for the Federal Republic of Germany, 1949 (Art. 2 Abs. 1 i. V. m. Art. 1 Abs. 1)

"'Article 2 - Personal freedoms'" (1) Every person shall have the right to free development of his personality insofar as he does not violate the rights of others or offend against the constitutional order or the moral law. Article 1 [Human dignity – Human rights – Legally binding force of basic rights] (1) Human dignity shall be inviolable. To respect and protect it shall be the duty of all state authority. https://www.btg-bestellservice.de/pdf/80201000.pdf Basic Law for the Federal Republic of Germany - Basic Law for the Federal Republic of Germany

"'Further: Article 1 and Article 10:'"

Art. 1 Human dignity – Human rights: Legally binding force of basic rights (1) Human dignity shall be inviolable. To respect and protect it shall be the duty of all state authority. (2) The German people therefore acknowledge inviolable and inalienable human rights as the basis of every community, of peace and of justice in the world. (3) The following basic rights shall bind the legislature, the executive and the judiciary as directly applicable law

16.6 Art. 10 - Privacy of correspondence, posts and telecommunications

Secrecy of correspondence - Fernmeldegeheimnis (Art. 10 Abs. 1 Grundgesetz)

§ 88 Abs. 1 Fernmeldegeheimnis - Telekommunikationsgesetz: (1) Dem Fernmeldegeheimnis unterliegen der Inhalt der Telekommunikation und ihre näheren Umstände, insbesondere die Tatsache, ob jemand an einem Telekommunikationsvorgang beteiligt ist oder war. Das Fernmeldegeheimnis erstreckt sich auch auf die näheren Umstände erfolgloser Verbindungsversuche. (2) Zur Wahrung des Fernmeldegeheimnisses ist jeder Diensteanbieter verpflichtet. Die Pflicht zur Geheimhaltung besteht auch nach dem Ende der Tätigkeit fort, durch die sie begründet worden ist. (3) Den nach Absatz 2 Verpflichteten ist es untersagt, sich oder anderen über das für die geschäftsmäßige Erbringung der Telekommunikationsdienste einschließlich des Schutzes ihrer technischen Systeme erforderliche Maß hinaus Kenntnis vom Inhalt oder den näheren Umständen der Telekommunikation zu verschaffen. Sie dürfen

Kenntnisse über Tatsachen, die dem Fernmeldegeheimnis unterliegen, nur für den in Satz 1 genannten Zweck verwenden. Eine Verwendung dieser Kenntnisse für andere Zwecke, insbesondere die Weitergabe an andere, ist nur zulässig, soweit dieses Gesetz oder eine andere gesetzliche Vorschrift dies vorsieht und sich dabei ausdrücklich auf Telekommunikationsvorgänge bezieht. Die Anzeigepflicht nach § 138 des Strafgesetzbuches hat Vorrang. (4) Befindet sich die Telekommunikationsanlage an Bord eines Wasser- oder Luftfahrzeugs, so besteht die Pflicht zur Wahrung des Geheimnisses nicht gegenüber der Person, die das Fahrzeug führt oder gegenüber ihrer Stellvertretung.

16.7 § 206 - Verletzung des Post- oder Fernmeldegeheimnisses

(1) Wer unbefugt einer anderen Person eine Mitteilung über Tatsachen macht, die dem Post- oder Fernmeldegeheimnis unterliegen und die ihm als Inhaber oder Beschäftigtem eines Unternehmens bekanntgeworden sind, das geschäftsmäßig Post- oder Telekommunikationsdienste erbringt, wird mit Freiheitsstrafe bis zu fünf Jahren oder mit Geldstrafe bestraft. (2) Ebenso wird bestraft, wer als Inhaber oder Beschäftigter eines in Absatz 1 bezeichneten Unternehmens unbefugt 1. eine Sendung, die einem solchen Unternehmen zur Übermittlung anvertraut worden und verschlossen ist, öffnet oder sich von ihrem Inhalt ohne Öffnung des Verschlusses unter Anwendung technischer Mittel Kenntnis verschafft, 2. eine einem solchen Unternehmen zur Übermittlung anvertraute Sendung unterdrückt oder 3. eine der in Absatz 1 oder in Nummer 1 oder 2 bezeichneten Handlungen gestattet oder fördert. (3) Die Absätze 1 und 2 gelten auch für Personen, die 1. Aufgaben der Aufsicht über ein in Absatz 1 bezeichnetes Unternehmen wahrnehmen, 2. von einem solchen Unternehmen oder mit dessen Ermächtigung mit dem Erbringen von Post- oder Telekommunikationsdiensten betraut sind oder 3. mit der Herstellung einer dem Betrieb eines solchen Unternehmens dienenden Anlage oder mit Arbeiten daran betraut sind. (4) Wer unbefugt einer anderen Person eine Mitteilung über Tatsachen macht, die ihm als außerhalb des Post- oder Telekommunikationsbereichs tätigem Amtsträger auf Grund eines befugten oder unbefugten Eingriffs in das Post- oder Fernmeldegeheimnis bekanntgeworden sind, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft. (5) Dem Postgeheimnis unterliegen die näheren Umstände des Postverkehrs bestimmter Personen sowie der Inhalt von Postsendungen. Dem Fernmeldegeheimnis unterliegen der Inhalt der Telekommunikation und ihre näheren Umstände, insbesondere die Tatsache, ob jemand an einem Telekommunikationsvorgang beteiligt ist oder war. Das Fernmeldegeheimnis erstreckt sich auch auf die näheren Umstände erfolgloser Verbindungsversuche.

- http://www.gesetze-im-internet.de/gg/art_10.html
- Secrecy of correspondence - Secrecy of correspondence
- Briefgeheimnis - Briefgeheimnis
- (Fernmeldegeheimnis - Fernmeldegeheimnis]()
- Postgeheimnis - Postgeheimnis
- http://www.gesetze-im-internet.de/tkg_2004/__88.html
- http://www.gesetze-im-internet.de/stgb/__206.html

16.8 United States Constitution: Search and Seizure (Expectation of Privacy, US Supreme Court)

The right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no Warrants shall issue, but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized. http://www.usconstitution.net/const.html

17 History of Program Publications

The list of publications shows continuous releases of the application over several years. The first publication dates back to 2013, and before that, another project also involved several years of research work. The release dates of the versions show on average almost a month a release. The note makes it clear which feature has been added, improved, or published.

The history of the publications since 2013 and earlier can be found with approx. 40 Program Releases here in the wiki of the project page: https://sourceforge.net/p/goldbug/wiki/release-history/

18 Website

Further information can be found on the website:

- http://GoldBug.sf.net

19 Open source code

The open source code can be found at GitHub:

- https://github.com/textbrowser/spot-on

19.1 Compile Information

Anyone who looks on the website of GoldBug, finds here the current release, especially for Windows. If you have advanced computer skills, would like to compile a program yourself from the source code or want to learn from this example, you will find here hints on how to proceed for the operating system Windows.

The compilation from the source code allows the user to see how the source code forms into a binary file (.exe) and which program libraries are to be supplemented so that the executable file can run.

1. First, download the Qt tool kit. To choose would be the offline (or online) installation of Qt with MingGW: eg Qt 5.X for Windows 32-bit (MinGW 4.9.2, 1.0 GB) at the URL: http://www.qt.io/download- open-source / # section-2

2. Then the source code is to be downloaded. For Windows all required dependencies and libraries are already integrated in the path of the source text. The GoldBug Gui and the Spot-On Kernel can be found at GitHub at this URL: https://github.com/textbrowser/spot-on To download the source code, the user can download the master tree on the website as a zip in the browser or use a GIT client (Windows) for source download.

For Linux, all these libraries are to be installed:

- Qt 5.1.x or higher,
- libGeoIP 1.5.1,
- libcrypto 0.9.8 or later,
- libgcrypt 1.5.x, and
- libssl 0.9.8 or later.
- libsqlite3-dev
- libgcrypt11-dev
- libssl-dev
- libgeoip-dev
- libpq-dev,
- libeay,
- libgpg-error,
- libsshgcrypt-dev,
- libssh-gcrypt-dev,
- libgcrypt-dev,
- libgcrypt11-dev,
- libgl1-mesa-dev,
- libcurlpp-dev,
- libcurl4openssl-dev,
- libsctp-dev,
- libtool,
- libtool-dev,
- libntl,

1. The libGeoIP program library is optional and can be bypassed if the selected Qt-PRO project file is configured accordingly. It has to be checked, whether for Linux all mentioned or more recent versions of these program libraries are installed on the machine. For Windows, the necessary program libraries are already attached to the source code (DLL files). Further compiling information can be found here: https://sourceforge.net/p/goldbug/wiki/compiling/

2. After the user has installed Qt, start the program Qt-Creator from the Qt directory.

Then select the relevant .pro file from the unpacked source code path and compile the GUI and the kernel with Qt Creator. For the compilation of GoldBug you install Qt5 and then select the .pro file “GoldBug.Qt5.win.pro”. This file opens both kernel and gui sub-pro files.

Then in QT-Creator simply click on the green forward arrow and start compiling. At the end of the compilation process from the Qt Creator GoldBug.exe should then be bootable. If the user wants to put the exe.file in a separate path on his harddisk, he has to add all needed DLL files (from the selected Qt version and from all current libraries) as well as the subpaths e.g. for the sound or Qt files , as they already exist in the default installation zip for GoldBug Windows. The library DLL files for Window are also stored in the source code of the respective library paths for convenient and easy use.

The user can of course compile with the Qt terminal window GoldBug also with manual DOS commands, without using Qt-Creator.

COMPILING PROCESS with C ++ / Qt:

- Source: https://github.com/textbrowser/spot-on

’’’Windows:’’’ qmake -o Makefile GoldBug.win.qt5.pro
make or mingw32-make
or choose in Qt-Creator: GoldBug.win.qt5.pro

GB does not provide checksums for the binary downloads as the source is given for those who want to build on their own. Please notice: GB has a build date in the gui so the sums might differ for each compile!

’’’FURTHER INFO for other .pro files: ’’’

If header (h) or interface (ui) files have changed, please perform a distclean before building the application.

’’’Absolute cleaning:’’’ make distclean or mingw32-make distclean

’’’FreeBSD:’’’ qmake -o Makefile spot-on.freebsd.pro
make

’’’Linux:’’’ qmake -o Makefile spot-on.pro
make

’’’OS X:’’’ qmake -spec macx-g++ -o Makefile spot-on.osx.pro
make

’’’Windows:’’’ qmake -o Makefile spot-on.win.pro
make or mingw32-make

20 List of publications

- Adams, David / Maier, Ann-Kathrin (2016): BIG SEVEN Study, open source crypto-messengers to be compared - or: Comprehensive Confidentiality Review & Audit of GoldBug, Encrypting E-Mail-Client & Secure Instant Messenger, Descriptions, tests and analysis reviews of 20 functions of the application GoldBug based on the essential fields and methods of evaluation of the 8 major international audit manuals for IT security investigations including 38 figures and 87 tables., URL: https://sf.net/projects/GoldBug/files/bigseven-crypto-audit.pdf - English / German Language, Version 1.1, 305 pages, June 2016

- Arbeitskreis Vorratsdatenspeicherung (AKV), Bündnis gegen Überwachung et al.: List of Secure Instant Messengers, URL: http://wiki.vorratsdatenspeicherung.de/List_of_Secure_Instant_Messengers, Mai 2014.

- Banerjee, Sanchari: EFYTIMES News Network: 25 Best Open Source Projects Of 2014: Efytimes ranked GoldBug Messenger # 4 on the overall Top 25 Best Open Source Projects Of 2014, http://www.efytimes.com/e1/fullnews.asp?edid=148831

- Cakra, Deden: Review of GoldBug Instant Messenger, Blogspot, 13 Desember 2014, http://bengkelcakra.blogspot.de/2014/12/free-download-GoldBug-instant-messenger.html

- Constantinos / OsArena: GoldBug: ΜΙΑ ΣΟΥΙΤΑ ΓΙΑ CHATING ΜΕ ΠΟΛΛΑΠΛΗ ΚΡΥΠΤΟΓΡΑΦΗΣΗ, Latest Articles, 25 March 2014, http://osarena.net/logismiko/applications/GoldBug-mia-souita-gia-chating-me-pollapli-kriptografisi.html

- Demir, Yigit Ekim: Güvenli ve Hizli Anlik Mesajlasma Programi: GoldBug Instant Messenger programi, bu sorunun üstesinden gelmek isteyen kullanicilar için en iyi çözümlerden birisi haline geliyor ve en güvenli sekilde anlik mesajlar gönderebilmenize imkan taniyor (Translated: “GoldBug Instant Messenger Application is the best solution for users, who want to use one of the most secure ways to send instant messages”), News Portal Tamindir http://www.tamindir.com/GoldBug-instant-messenger/

- Dragomir, Mircea: GoldBug Instant Messenger - Softpedia Review: This is a secure P2P Instant Messenger that ensures private communication based on a multi encryption technology constituted of several security layers, http://www.softpedia.com/get/Internet/Chat/Instant-Messaging/GoldBug-Instant-Messenger.shtml, Softpedia Review, January 31st, 2016

- Edwards, Scott: Manual of the GoldBug Crypto Messenger, 2018, Review at Github, URL: https://compendio.github.io/goldbug-manual-de/ & https://compendio.github.io/goldbug-manual

- Filecluster: GoldBug Instant Messenger - Un programme très pratique et fiable, conçu pour créer un pont de communication sécurisé entre deux ou plusieurs utilisateurs, URL: https://www.filecluster.fr/logiciel/GoldBug-Instant-Messenger-174185.html

- Fousoft: GoldBug Instant Messenger, URL: https://www.fousoft.com/goldbug-instant-messenger.html, March 16, 2017
- Generation NT: Sécuriser ses échanges par messagerie: Apportez encore plus de la confidentialité dans votre messagerie, URL: https://www.generation-nt.com/goldbug-messenger-securiser-echanger-communiquer-discuter-messagerie-securite-echange-communication-telecharger-telechargement-1907585.html
- Hartshorn, Sarah: 3 New Open Source Secure Communication Projects, May 28, 2015, http://blog.vuze.com/2015/05/28/3-new-open-source-secure-communication-projects/
- Harvey, Cynthia: Datamation: 50 Noteworthy Open Source Projects - Chapter Secure Communication: GoldBug Messenger ranked on first # 1 position, Posted September 19, 2014, http://www.datamation.com/open-source/50-noteworthy-new-open-source-projects-3.html
- Heise: GoldBug kann Schlüssel selbst encodiert versenden, URL: http://www.heise.de/download/goldbug-1192605.html
- Joos, Thomas: Sicheres Messaging im Web, PCWelt Magazin, Mittwoch den 01.10.2014, http://www.pcwelt.de/ratgeber/Tor__I2p__Gnunet__RetroShare__Freenet__GoldBug__Spurlos_im_Web-Anonymisierungsnetzwerke-8921663.html
- Lindner, Mirko: POPTASTIC: Verschlüsselter Chat über POP3 mit dem GoldBug Messenger, Pro-Linux, 9. Dezember 2014, http://www.pro-linux.de/news/1/21822/POPTASTIC-verschluesselter-chat-ueber-pop3.html
- Majorgeeks:GoldBug Secure Email Client & Instant Messenger, URL:http://www.majorgeeks.com/files/details/goldbug_secure_email_client_instant_messenger.html
- Momedo: Open Source Mobiler Messenger für kommunale und schulische Zwecke mit Verschlüsselung, Github, URL: https://momedo.github.io/momedo/ & https://github.com/momedo/momedo/blob/master/README.md , 2018
- Por, Julianna Isabele: Segurança em primeiro lugar, URL: https://www.baixaki.com.br/download/goldbug.htm
- Qt Digia: Qt Digia has awarded GoldBug IM as reference project for Qt implementation in the official Qt-Showroom of Digia: showroom.qt-project.org/goldbug/, 2015
- Sabtu: Free GoldBug Instant Messenger 1.7, URL: http://bengkelcakra.blogspot.de/2014/12/free-download-goldbug-instant-messenger.html, 13 December 2014
- Schneier, Bruce / Seidel, Kathleen / Vijayakumar, Saranya: GOLDBUG Multi-Encrypting Messenger – in: A Worldwide Survey of Encryption Products, URL:

https://www.schneier.com/cryptography/paperfiles/worldwide-survey-of-encryptionproducts.pdf, February 11, 2016 Version 1.0.

- Security Blog: Secure chat communications suite GoldBug. Security Blog, 25. März 2014, http://www.hacker10.com/other-computing/secure-chat-communications-suite-GoldBug/
- Spot-On (2014): Documentation of the Spot-On-Application, URL: https://github.com/textbrowser/spot-on/tree/master/branches/trunk/Documentation, Github 2014.
- Spot-On (2011): Documentation of the Spot-On-Application, URL: https://sourceforge.net/p/spoton/code/HEAD/tree/, under this URL since 06/2013, Sourceforge, including the Spot-On: Documentation of the project draft paper of the pre-research project since 2010, Project Ne.R.D.D., Registered 2010-06-27, URL: https://sourceforge.net/projects/nerdd/ has evolved into Spot-On. Please see http://spot-on.sf.net and URL: https://github.com/textbrowser/spoton/blob/master/branches/Documentation/RELEASE-NOTES.archived, 08.08.2011.
- Theisen, Michaela: GoldBug Instant Messenger - Beliebte Software, Sicherer Instant Messenger, URL: https://www.freeware-base.de/freeware-zeige-details-28142-GoldBug_Instant_Messenger.html, 2015
- Tur, Henryk / Computerworld: GoldBug Secure Email Client & Instant Messenger, https://www.computerworld.pl/ftp/goldbug-secure-email-client-instant-messenger.html, 11.01.2018
- Vaughan-Nichols, Steven J.: How to recover from Heartbleed, ZDNet, April 9, 2014, http://www.zdnet.com/how-to-recover-from-heartbleed-7000028253
- Weller, Jan: Testbericht zu GoldBug für Freeware, Freeware-Blog https://www.freeware.de/download/GoldBug/

Further bibliographic hints:

- Bernat, V. (2012, January 1). SSL/TLS & Perfect Forward Secrecy. Web article retrieved March 5, 2015, from http://vincent.bernat.im/en/blog/2011-sslperfect- forward-secrecy.html;
- Schum, Chris: Correctly Implementing Forward Secrecy, SANS Institute InfoSec Reading Room, March 14, 2014,
- Zhu, Y. (2014, April 8): Why the Web Needs Perfect Forward Secrecy More Than Ever. Web article. Retrieved February 2, 2015, from https://www.eff.org/deeplinks/2014/04/why-web-needs-perfect-forward-secrecy

goldbug-manual is maintained by compendio.

This page was generated by GitHub Pages.